DELTA

BASIC

VOOR MSX COMPUTERS

GARANTIE EN SERVICE (*)

Aan geregistreerde gebruikers van Delta Basic geeft Filosoft een tamelijk vergaande garantie en service op het programma en de bijbehorende handleiding. Een en ander geldt tot twee jaar na de aankoopdatum van Delta Basic.

Garantie:

- Indien er tijdens de fabricage of het transport van dit exemplaar van Delta Basic een beschadiging aan de diskette of cassette en/of aan de handleiding mocht zijn opgetreden, die het werken met het programma ernstig bemoeilijkt, kunt u de disk/cassette en/of de handleiding opsturen naar Filosoft, en we zenden u per kerende post kosteloos een nieuw exemplaar toe.

- Indien een beschadiging optreedt in de periode dat Delta Basic in gebruik is, wordt eveneens de hierboven genoemde procedure gevolgd, met dien verstande, dat u bij uw verzoek om een vervangingsexemplaar f 7,50 aan postzegels dient bij te sluiten voor verpakkings- en verzendkosten.

Service:

- Gebruikers van Delta Basic die advies behoeven over het programma kunnen iedere VRIJDAG tijdens kantooruren gratis gebruik maken van de Filosoft-servicelijn (050-137746).

- Indien dit programma zodanig herzien en/of uitgebreid wordt dat sprake is van een nieuwe release, worden geregistreerde gebruikers daarvan op de hoogte gesteld en kunnen zij tegen sterk gereduceerde prijs een geheel nieuwe versie ontvangen bij inruil van de versie in hun bezit.

(*) N.B.:

Een gebruiker van Delta Basic kan alleen aanspraak maken op bovengenoemde garantie en service (ook de telefonische adviezen) als deze gebruiker als zodanig bij Filosoft geregistreerd staat.
Bij deze handleiding wordt daartoe verstrekt een servicekaart, die u zo spoedig mogelijk na aankoop van het programma dient op te sturen:

Filosoft
Postbus 1353
9701 BJ Groningen

INHOUDSOPGAVE

INTRODUKTIE 5

vooraf 5
laden 5
windows 5
procedures 6
chainen 6
utilities 7
geheugengebruik 8
stoppen 8

BESCHRIJVING COMMANDO'S EN FUNKTIES 9

averify 9
baud 10
biprint 11
biset 12
bverify 13
capson/capsoff 14
chain 15
clearsprite 16
cls# 17
codetodata 18
common 19
coplin 20
defproc 21
deleteproc 22
dfiles/ldfiles 23
dpeek 24
dpoke 25
endproc 26
fill# 27
find(next/all)/lfind(next/all) 28
get 29
inifnk 30
inipsg 31
killbuf 32
killrem/lkillrem 33
listdata/llistdata 34
listgo/llistgo 35
listlin/llistlin 36
listproc/llistproc 37
listpsg/llistpsg 38
listscreen/llistscreen 39
listtype/llisttype 40
listusr/llistusr 41
listvar/llistvar 42
listwindow/llistwindow 43
locate# 44
lower 45
mergeproc 46
movlin 47
pause 48
pol 49
proc 50

INHOUDSOPGAVE (VERVOLG)

quit 51
replace(next/all)/lreplace(next/all) 52
rolld/rolll/rollr/rollu 53
saveproc 54
screendump 55
screenon/screenoff 56
screensave 57
setdrive 58
status/lstatus 59
storescreen/restorescreen 60
tfiles/ltfiles 61
unnew 62
upper 63
wbox# 64
window# 65
winput# 66
wprint# 67
wrapd/wrapl/wrapr/wrapu 68

BIJLAGEN 69

bijlage A - bit image printen 69
bijlage B - overzicht (fout)meldingen 70

INTRODUKTIE

VOORAF

Met het programma Delta Basic worden de grenzen van MSX-BASIC opnieuw verlegd.
Hoewel MSX-BASIC op zich al het resultaat was van jarenlange ontwikkeling en verbeteringen in BASIC-dialecten, en bij het uitbrengen van de MSX2-computers daar nog een schepje bovenop werd gedaan, bleek in de praktijk dat vele gebruikers behoefte hadden aan extra commando's.
Wie een tijdje de computerbladen volgt, ziet hoe cratief programmeurs zijn in het zelf bedenken van oplossingen voor problemen, die eigenlijk niet binnen MSX-BASIC kunnen worden opgelost. Vaak worden daartoe korte stukjes machinetaal-programma's aan het BASIC toegevoegd.

Met Delta Basic worden de mogelijkheden van het programmeren in BASIC enorm uitgebreid. Daartoe staan de programmeur bijna 100 nieuwe commando's en functies tot zijn beschikking, die na enige gewenning net zo gemakkelijk gebruikt worden als alle 'normale' BASIC-commando's.
Eenieder die enige tijd met Delta Basic werkt, zal zich afvragen hoe hij eigenlijk altijd zonder gekund heeft. Daarom is Delta Basic nu reeds op weg om de facto een nieuwe standaard te worden door de BASIC-programmeur die een MSX gebruikt.

LADEN

Na het aanzetten van de computer dient u eerst Delta Basic te laden alvorens u begint met programmeren of een ander programma wilt laden.
Het programma wordt geladen door te typen:

RUN "DELTA"	**<RETURN>**	bij gebruik van een diskette
RUN "CAS:DELTA"	**<RETURN>**	bij gebruik van een cassette

Zodra Delta Basic geladen is verschijnt op het beeldscherm de melding 'Delta Basic version ...'
Delta Basic wordt in de 'achtergrond' van het computergeheugen gezet. Het blijft aanwezig zolang u de computer niet uitzet.

WINDOWS

Delta Basic biedt u de unieke mogelijkheid het beeldscherm in te delen in maximaal 9 windows.
Dit vergroot niet alleen de lay-outmogelijkheden op het scherm, doch maakt tevens het programmeren overzichtelijker.

Een window is een venster binnen het beeldscherm. Elk window afzonderlijk gedraagt zich alsof het een compleet beeldscherm was. Dit betekent het gebruik van PRINT-, INPUT-, LOCATE- en CLS- opdrachten binnen elk gewenst window. Het betekent ook bijvoorbeeld het automatisch omhoog rollen van een window, zodra de tekstruimte daarbinnen 'vol' is, net als bij het complete beeldscherm.
Met windows kunt u verschillende programma-taken een vaste ruimte op het beeldscherm geven. Met een simpel commando kan een window desge-

wenst van een kader worden voorzien.
Een bijzondere gebruiksmogelijkheid is het rollen van windows in elke gewenste richting, al dan niet met omslag.
Alle window-commando's werken in de SCREEN-modes 0, 1 en 2, zodat nu ook PRINT- en INPUT-commando's in SCREEN 2 zeer eenvoudig zijn geworden. Hoewel hogere SCREEN-modes niet gebruikt kunnen worden voor windows, zijn ze verder wel normaal te gebruiken onder Delta Basic.
Window-nummer 0 (nul) is vast gedefinieerd als het gehele beeldscherm. Daardoor kunnen ook bijvoorbeeld de rol-opdrachten op het complete beeldscherm worden toegepast.

Bij alle window-opdrachten wordt steeds achter een #-teken aangegeven op welk window de opdracht betrekking heeft. Met LISTWINDOW wordt een overzicht gegeven welke windows hoe gedefinieerd zijn op dat moment.

De commando's die op windows betrekking hebben zijn: WINDOW#, WPRINT#, WINPUT#, LOCATE#, CLS#, FILL#, ROLL, WRAP en LISTWINDOW.

PROCEDURES

Een van de nadelen van het programmeren in BASIC is, dat het snel onoverzichtelijk kan worden door het veelvuldig gebruik van GOTO's en GOSUB's.
Delta Basic biedt hiervoor een oplossing door het gebruik van procedures aan te bieden.
Een procedure bestaat uit een stukje BASIC dat begin met een DEFPROC en de naam van de procedure, en eindigt met een ENDPROC.
Vanuit ergens anders in het BASIC-programma wordt een procedure aangeroepen door het commando PROC met de naam van de gewenste procedure.
Procedures kunnen op een willekeurige plek in het programma staan, en bij het gebruik van PROC is alleen de naam van belang, geen regelnummers. Het is dus niet meer nodig te onthouden waar ook al weer ergens een bepaalde subroutine stond.

Bijzonder handig is de mogelijkheid om procedures te saven en later weer terug te halen via de commando's SAVEPROC en MERGEPROC.
Daarmee kan een procedure die in een programma gemaakt is, en wellicht ook in andere programma's handig kan zijn, weggeschreven worden om later toe te voegen aan een ander programma.
In feite kan hiermee een complete bibliotheek van procedures worden aangemaakt.

De commando's DELETEPROC en LISTPROC completeren de gebruiksmogelijkheden van procedures, en spreken verder voor zich.

CHAINEN

Bij MSX-computers is de geheugenruimte die voor het programmeren in BASIC ter beschikking staat, relatief gering. Bij een computer zonder diskdrive of bij uitgeschakelde drive is bijna 29KByte beschikbaar; zodra een diskdrive aanwezig is, wordt dit met enkele duizenden bytes verkleind.
Veel gebruikers zijn met de grenzen van het geheugen in aanraking gekomen door een plotselinge melding 'Out of memory'. Waarna tal van kunstgrepen toegepast moesten worden om een programma toch lopend te krijgen in de beschikbare ruimte.

Hoewel een computer op zich nog veel meer geheugen kan bevatten (soms wel 256K), is dit niet vanuit BASIC aanspreekbaar.

Delta Basic biedt nu een totaal nieuwe mogelijkheid om toch zeer lange programma's te kunnen maken en gebruiken.
De mogelijkheid bestond al om vanuit een programma een ander programma in te laden en te runnen (via RUN"naam"); alleen waren dan wel alle variabelen-gegevens verdwenen. Met Delta Basic kunt u nu een ander programma inladen en runnen met behoud van variabelen!

Het gebruik is simpel: met COMMON wordt aangegeven welke variabelen 'meegenomen' moeten worden naar een volgend programma; met CHAIN wordt de opdracht gegeven het volgende programma te laden, de variabelen weer op de goede plaats te zetten, en te beginnen met de uitvoering van het nieuwe programma.

Bij gebruik van een cassetterecorder zal het in de praktijk alleen zinvol zijn een programma te CHAIN-en dat verderop op het bandje staat. Bij gebruik van een diskdrive echter kunnen programma's elkaar in alle denkbare volgordes aanroepen.
Daarmee is een in principe oneindig groot programma mogelijk, dat is opgebouwd uit verschillende modules. Die modules zijn dan als aparte programma's gesaved, en komen één voor één in het geheugen terecht; toch kunnen uitkomsten uit een module via variabelen doorgegeven worden aan een volgende module.

UTILITIES

Hierboven zijn de meest wezenlijke uitbreidingen op het normale MSX-BASIC genoemd, die Delta Basic u biedt.
Daarnaast kent Delta Basic tientallen verschillende nieuwe commando's en functies die het programmeren nog eens extra gemakkelijk en overzichtelijk kunnen maken, of mogelijkheden bieden die tot nu toe slechts met behulp van machinetaal-routines denkbaar waren (zelf gemaakt of uit een tijdschrift overgenomen).

We noemen hier de uitgebreide LIST-mogelijkheden, voor overzichten van variabelen, GOTO- en GOSUB-gebruik, procedures, windows, etc.
Daarnaast de commando's COPLIN en MOVLIN om stukken BASIC te copiëren of te verplaatsen, FIND en REPLACE om iets in een programma te vinden en/of te vervangen.
Er zijn commando's mogelijk om te allen tijde een screendump op de printer te maken, en om de complete MSX-karakterset ook op niet-MSX-printers te gebruiken.
De gebruiker van een cassetterecorder komt aan zijn trekken door nieuwe commando's voor het verifiëren van bestanden, voor het geven van een overzicht van de cassette-inhoud, of voor het instellen van andere BAUD-rates.
Wellicht spreken andere commando's en functies zoals CAPSON, LOWER, UPPER, DPOKE, DPEEK, PAUSE, POL, GET, SETDRIVE en UNNEW direct tot de verbeelding; anders zult u na het lezen van hun beschrijving snel de mogelijkheden ervan inzien.

Alle nieuwe commando's en functies zijn niet in deze inleiding genoemd. Een compleet overzicht vindt u in de rest van deze handleiding, waarbij tevens behandeld worden de juiste schrijfwijze, de precieze werking, eventuele (fout)meldingen die kunnen voorkomen, tips bij het gebruik alsmede vele gebruiksvoorbeelden.

GEHEUGENGEBRUIK

Wanneer Delta Basic is ingeladen, verplaatst het zichzelf naar een stuk geheugen dat normaal niet vanuit BASIC toegankelijk is. Van de voor BASIC beschikbare geheugenruimte neemt Delta Basic slechts enkele bytes af.

Commerciële programma's maken vaak van hetzelfde stuk geheugen gebruik waar Delta Basic zit. Gelijktijdig gebruik kan in zo'n geval leiden tot een 'hang-up' van de computer. Delta Basic is natuurlijk ook niet bedoeld om commerciële machinetaal-programma's te verbeteren.

Memory disk is een optie die op MSX2-computers geboden wordt. Ook deze kan conflicten geven met Delta Basic. Daartoe is in Delta Basic zelf een beveiliging ingebouwd: bij het geven van een commando CALL MEMINI wordt u verplicht een hoeveelheid ruimte op te geven tot maximaal 16000 bytes.
Commando's in Delta Basic die een reeds gebruikte memory disk zouden kunnen overschrijven, geven in dergelijke gevallen een waarschuwing.

MSX-DOS gebruikt tevens dezelfde geheugenruimte als Delta Basic. U kunt wel vanuit MSX-DOS als gebruikelijk Delta Basic inladen door na de prompt A> in te typen: BASIC DELTA. Daarna mag u echter niet meer het commando CALL SYSTEM gebruiken.

Delta Basic zelf is gesitueerd in page 1 van de RAM (4000-7FFF); page 0 (0-3FFFF) wordt gebruikt voor tijdelijke opslag bij STORESCREEN en CHAIN. Delta Basic maakt tevens veelvuldig gebruik van zogenaamde HOOKS.

STOPPEN

Wenst u te stoppen met het werken in Delta Basic, gebruik dan het commando QUIT, of zet de computer enige tijd uit.

AVERIFY *Verifieer naar cassette weggeschreven ASCII BASIC files*

SYNTAX : AVERIFY "[bestandsnaam]"

Bestandsnaam is een rij van maximaal 6 lettertekens, waarbij hoofd- en kleine letters een onderscheiden betekenis hebben.
Bij weglaten bestandsnaam wordt het eerstvolgende bestand geverifieerd.

WERKING:

Het op dat moment in het geheugen van de computer aanwezige programma wordt vergeleken met een programma dat onder de opgegeven naam op een cassette is opgeslagen. Een dergelijk programma dient weggeschreven te zijn in het zogenaamde ASCII-formaat, dus met het commando SAVE "CAS: bestandsnaam".
Tijdens het verifiëren wordt het van cassette gelezen programma tegelijk afgebeeld op het beeldscherm.
Als de programma's niet overeenstemmen zal de melding "Verify error!" verschijnen; anders komt de melding "Verified OK!".
Het commando kan afgebroken worden door CONTROL+STOP te drukken.

AANVERWANTE COMMANDO'S: CLOAD?, BVERIFY

(FOUT)MELDINGEN:

Verified OK!	- het programma staat correct op cassette
Verify error	- het programma staat niet correct op cassette
MODE error	- het commando dient direct gegeven te worden (niet in een programma)
Skip:	- de naam van een gevonden bestand komt niet overeen met de opgegeven naam; er wordt verder gezocht

TIPS:

Denk eraan dat het BASIC programma gesaved moet zijn met SAVE"CAS:bestandsnaam".
Het commando AVERIFY is alleen beschikbaar in de "command mode", en kan dus niet in een programma worden opgenomen.
Indien regelmatig verify-fouten optreden, verlaag dan eventueel de snelheid waarmee gesaved wordt middels het nieuwe BAUD-commando.

gebruiksvoorbeeld:

AVERIFY "PROG2"

BAUD *Selecteer een SAVE-snelheid naar cassette*

SYNTAX : BAUD snelheid

Snelheid is een numerieke constante, variabele, lijstvariabele of uitdrukking daarmee.
Toegestane waarden zijn: 900,1200,1500,1800,2100,2400,2700 en 3000.

WERKING:

Het wegschrijven van een programma of stuk code naar cassette gebeurt met een vast ingestelde snelheid (opgegeven in BAUD = aantal bits per seconde). In normaal MSX-BASIC zijn daarbij slechts de waarden 1200 en 2400 BAUD mogelijk, in te stellen met het SCREEN-commando.
Het BAUD-commando geeft u de mogelijkheid om een zo hoog mogelijke snelheid te kiezen voor uw eigen recorder. Bij het laden van een programma is het niet nodig de snelheid op te geven: de computer stelt zich automatisch goed in.
De ingestelde waarde kan opgevraagd worden door middel van het nieuwe LISTSCREEN-commando.

AANVERWANTE COMMANDO'S: CSAVE, SAVE, BSAVE, SCREEN
CLOAD, LOAD, BLOAD, LIST SCREEN

(FOUT)MELDINGEN:

Illegal function call - de opgegeven waarde is niet mogelijk

TIPS:

De MSX-computer is vrij kritisch wat betreft de kwaliteit van opnamen op cassettes.
Een recorder met schone en goed afgestelde koppen is erg belangrijk.
Met het BAUD-commando kunt u desgewenst belangrijke programma's of bestanden extra 'veilig' saven door een lagere snelheid te kiezen.
Andersom kunt u uitproberen wat de grootst mogelijke snelheid is die uw recorder zonder problemen aankan.

gebruiksvoorbeeld:

```
BAUD 1800
```

BIPRINT *Zet de printer in bit-image mode*

SYNTAX : BIPRINT (breedte[,optie[,optie[,optie[,optie]]]])
BIPRINT OFF

Breedte is een geheel getal tussen 1 en 8 dat aangeeft met hoeveel punten (pixels) een karakter geprint moet worden.
Optie kan zijn D,U,I en/of M; meerdere opties tegelijk zijn mogelijk.

- **D = dubbelbreed**
- **U = upside (soms juist normaal)**
- **I = invers**
- **M = mirror (spiegelbeeld)**

WERKING:

BIPRINT zorgt ervoor dat te printen karakters na commando's als LPRINT of LLIST in de zogenaamde bit-image mode geprint worden.
Het commando geeft de mogelijkheid de komplete MSX-karakterset ook op niet-MSX-printers te gebruiken. Daarnaast zijn een aantal opties ingebouwd, die elke dot-matrix printer enkele nieuwe mogelijkheden biedt.
Voordat dit commando goed kan werken moet in het algemeen eerst het nieuwe BISET-commando gegeven worden!
De karakterbreedte (het aantal bits dat per karakter gebruikt wordt) mag variëren van 1 t/m 8, maar de waarden 6 en 8 zijn het meest geschikt. Bij sommige printers zal blijken dat alle karakters op zijn kop op het papier komen te staan. In dat geval zal bij elk BIPRINT-commando de optie U gebruikt moeten worden. Een combinatie van de extra opties is mogelijk, bv. BISET (8,D,I) om karakters van 8 breed, dubbelbreed en invers af te beelden.
Elk te printen karakter wordt eerst in een interne buffer opgeslagen. Zodra een teken met een ASCII-code tussen 2 en 31 aan de orde is (bijvoorbeeld de code voor een nieuwe regel) vindt het feitelijke printen plaats.
Het bit-image printen wordt uitgeschakeld door het commando BIPRINT OFF te geven.
LET OP!! Bij een MSX-printer standaard de optie ',U' gebruiken !!

AANVERWANTE COMMANDO'S: BISET,LPRINT,LLIST

gebruiksvoorbeeld:

```
10 ' GEBRUIK BIJ EEN NIET-MSX PRINTER EERST HET BISET COMMANDO
20 LPRINT"BI-PRINT DEMO":LPRINT
30 BIPINT(8):' BIJ MSX-PRINTERS  BIPRINT(8,U) !!
40 LPRINT "lle MSX karakters"
50 FOR N=0 TO 255
60 IF N<32 THEN LPRINT CHR$(1);CHR$(N+64);
70 IF N>=32 THEN LPRINT CHR$(N);
80 NEXT N
90 BIPRINT(8,D):LPRINT "DUBBEL breed"
100 BIPRINT(8,U):LPRINT "OP ZYN KOP of juist normaal"
110 BIPRINT(8,I):LPRINT "INVERSE karakters"
120 BIPRINT(8,D,I):LPRINT "COMBINATIES zijn ook mogelijk"
130 BIPRINT OFF
```

BISET ***Leg de codes vast die de printer nodig heeft voor bit-image***

SYNTAX : BISET (codereeks)

Codereeks is een reeks van maximaal 12 gehele getallen (0-255), onderling gescheiden door komma's.

WERKING:

Teneinde een printer in de bit-image mode te zetten, heeft deze een aantal zogenaamde stuurcodes nodig. Om een correcte werking van de nieuwe BIPRINT- en SCREENDUMP-commando's mogelijk te maken moeten de benodigde stuurcode's eenmalig opgegeven worden. Daartoe dient het BISET-commando.
Een uitgevoerd BISET-commando blijft geldig tot een nieuw BISET-commando of totdat u DELTA BASIC verlaat.

Voorafgaande aan de codes voor de instelling van de bit-image mode kunnen desgewenst nog codes gezet worden die de regelafstand wijzigen; dit kan vooral belangrijk zijn ten behoeve van het SCREENDUMP-commando.

In Bijlage A treft u aanwijzingen aan hoe de juiste codes te vinden om te gebruiken in het commando BISET.
Heeft u de codes voor uw printer gevonden, en de correcte werking van het BISET-commando geconstateerd, dan is het raadzaam het BISET-commando met de codes in een procedure vast te leggen en te saven (SAVEPROC); deze procedure kan dan altijd eenvoudig aan een ander programma worden toegevoegd, zonder dat de juiste instelling opnieuw 'ontdekt' hoeft te worden.

AANVERWANTE COMMANDO'S: BIPRINT,LPRINT,LLIST,SCREENDUMP

TIPS:

In Bijlage A treft u de codes aan voor enkele soorten printers, alsmede aanwijzingen hoe u de instelling voor andere printers kunt bepalen.
Vaak zijn er per printer meerdere mogelijkheden voor bit-image, bijvoorbeeld 'enkele dichtheid' of 'dubbele dichtheid'. Experimenteer met deze mogelijkheden en kies degene die u zelf het beste toelijkt.

gebruiksvoorbeeld:

```
10 ' ALLEEN VOOR EPSON-PRINTERS !!!
20 '
30 BISET(&H1B,ASC("*"),0,192,3)
40 BIPRINT(8)
50 LPRINT"lle MSX tekens:"
60 FOR N=0 TO 255
70 IF N<32 THEN LPRINT CHR$(1);CHR$(N+64);
80 IF N>=32 THEN LPRINTCHR$(N);
90 NEXT N
100 BIPRINT OFF
```

BVERIFY *Verifieer naar cassette weggeschreven bytes*

SYNTAX : BVERIFY "[bestandsnaam]"

Bestandsnaam is een rij van maximaal 6 lettertekens, waarbij hoofd- en kleine letters een onderscheiden betekenis hebben.
Bij weglaten bestandsnaam wordt het eerstvolgende bestand geverifieerd.

WERKING:

Een bestand van bytes dat onder de opgegeven naam op cassette is opgeslagen (na een BSAVE-commando) wordt vergeleken met de geheugeninhoud van de computer.
Als alles overeenkomt worden van het bestand het beginadres, eindadres en opstartadres weergegeven alsmede de melding "Verified OK!".
Wordt echter een fout ontdekt, dan volgt de melding "Error at nn", waarbij 'nn' het geheugenadres is waarvan de inhoud niet overeenkomt met hetgeen op de cassette staat.
Het commando kan afgebroken worden door CONTROL+STOP te drukken.

AANVERWANTE COMMANDO'S: CLOAD?, AVERIFY

(FOUT)MELDINGEN:

Verified OK!	- de bytes op cassette zijn identiek aan die in het geheugen
Error at nn	- er is bij de verificatie een fout ontdekt bij adres nn
Mode error	- het commando dient direct gegeven te worden (niet in een programma)
Skip:	- de naam van een gevonden bestand stemt niet overeen met de opgegeven naam; er wordt verder gezocht

TIPS:

Dit commando is alleen beschikbaar in de "command mode", en kan dus niet in een programma worden opgenomen.
Indien regelmatig verify-fouten optreden, verlaag dan eventueel de snelheid waarmee gesaved wordt middels het nieuwe BAUD-commando.

gebruiksvoorbeeld:

```
BSAVE"CAS:TEST",50000,50250:PRINT"SPOEL TERUG !":BVERIFY""
```

CAPS ON *Zet de computer in HOOFD LETTER mode*
CAPS OFF *Zet de computer in kleine letter mode*

SYNTAX : CAPS ON
CAPS OFF

WERKING:

Dit commando schakelt uw MSX in HOOFDLETTER of in kleine letter mode. Wanneer de hoofdletter mode geselecteerd wordt zal het lampje van de hoofdlettertoets (CAPS LOCK) gaan branden. Met dit commando kunt u van te voren bepalen of b.v. een input in hoofd of kleine letters verkregen wordt.

AANVERWANTE COMMANDO'S: UPPER, LOWER, GET, INPUT

TIPS:

Ondanks het feit dat een bepaalde lettermode is geselecteerd kan een gebruiker tijdens de input handmatig omschakelen op een andere mode. Wilt u honderd procent zeker zijn dat alles hetzij in hoofdletters hetzij in kleine letters komt te staan, gebruik dan de nieuwe commando's UPPER respectievelijk LOWER.

gebruiksvoorbeeld:

```
10 ' zet hoofdletters aan
20 CAPS ON
30 PRINT"HET HOOFDLETTER LAMPJE (CAPS) IS NU AAN !"
40 PRINT"EEN INPUT ZAL NU EEN HOOFDLETTER STRING OPLEVEREN"
50 INPUT"TYPE EEN WOORD EN DRUK <RETURN>";A$
60 PRINT"HET WOORD WAS ";A$
70 ' zet hoofdletters uit
80 CAPS OFF
90 PRINT"HET HOOFDLETTER LAMPJE (CAPS) IS NU UIT !"
100 PRINT"EEN INPUT ZAL NU EEN kleine LETTER STRING OPLEVEREN"
110 INPUT"TYPE EEN WOORD EN DRUK <RETURN>;A$
120 PRINT"HET WOORD WAS ";A$
```

CHAIN *Laad en run een BASIC-programma met behoud van variabelen.*

SYNTAX : CHAIN "[dev:]bestandsnaam"

Dev is de aanduiding voor het opslagmedium (CAS, A-H of MEM), bestandsnaam is een rij lettertekens met een maximum van 6 (cassetterecorder) of 11 (bij diskdrive); dev mag eventueel weggelaten worden (vergelijk de regels bij de normale SAVE- en LOAD-opdrachten).

WERKING:

Laadt een nieuw BASIC programma in het geheugen, en bewaart daarbij alle waarden van de variabelen, strings, en arrays van het vorige gebruikte BASIC programma. Zodra het nieuwe programma geladen is, zal begonnen worden met de uitvoering van dat programma.
Zodra een CHAIN-commando gegeven is worden eerst alle aanwezige variabelen, en arrays opgeslagen in een normaal niet gebruikt deel van de computer. Hiervoor is 16K geheugen beschikbaar (kan ook al door MEM-DISK in gebruik zijn!). Is dat niet genoeg, dan kan het aantal variabelen en arrays gereduceerd worden d.m.v. het nieuwe COMMON-commando.

AANVERWANTE COMMANDO'S: COMMON, RUN

(FOUT)MELDINGEN:

TOO MUCH varspace	- er wordt meer dan 16K voor variabelen gebruikt
NO ROOM for vars	- het te chainen programma is ingeladen, maar is te lang om de oude variabelen te herbergen
OUT of string space	- de "string space" is te klein, gebruik CLEAR
MEMDISC initialised	- de memory-disk dreigt overschreven te worden

TIPS:

Bedenk goed dat zodra dit commando wordt uitgevoerd het oude programma uit het geheugen gewist wordt. Voordat u een programma met daarin een CHAIN-commando test, moet dit dus eerst weggeschreven worden!
De strings worden op een bepaalde manier bewaard. Wanneer een string in een BASIC-tekstregel gedefinieerd is (b.v. 10 A$="TEKST") dan wordt deze normaal niet opgeslagen in het "string geheugen". Het CHAIN-commando doet dat echter wel, zodat dat geheugendeel sneller vol kan raken.

gebruiksvoorbeeld:

```
[prog1]
10 A=50:A$="HALLO" :DIMB(1):DIMB$(1):B(0)=5:B(1)=10
20 B$(0)="STRING1":B$(1)="STRING2"
30 CHAIN"prog2"

[prog2]
10 PRINT"A     =";A:PRINT"A$    =";A$
20 PRINT"B(0) =";B(0):PRINT"B(1) =";B(1)
30 PRINT"B$(0)=";B$(0):PRINT"B$(1)=";B$(1)
```

CLEAR SPRITE *Wis sprites van scherm en uit geheugen (VRAM).*

SYNTAX : CLEAR SPRITE

WERKING:

Omdat een CLS de sprites op het scherm laat staan is deze functie een aanvulling om het scherm leeg te maken.
Alle sprite patronen worden gewist uit de VIDEO-RAM, alle spritekleuren worden gezet als voorgrondkleur, alle verticale posities worden gezet op 209 (buiten het scherm), en alle spritenummers worden gezet als hun vlaknummer.

AANVERWANTE COMMANDO'S: CLS, SCREEN

gebruiksvoorbeeld:

```
10 CLS:SCREEN 1,1
20 FOR N=0 TO 7
30  VPOKE(BASE(9))+N,255
40 NEXT N
50 '
60  FOR N=0 TO 31
70   PUTSPRITE N,(N*4,N*4),10,0
80 NEXT N
90 '
100 PRINT"  NU VOLGT EEN CLS"
110 PAUSE 2:CLS:PAUSE 2
120 PRINT"  EN NU EEN CLEARSPRITE"
130 PAUSE 5
140 CLEARSPRITE
```

CLS # *Wis de inhoud van een window.*

SYNTAX : CLS #window

Window is het nummer van een tevoren gedefinieerd window, waarbij moet gelden: 0 <= nummer <= 9 .

WERKING:

Maakt het scherm schoon binnen het opgegeven window.

AANVERWANTE COMMANDO'S: WINDOW#, FILL#, CLS

(FOUT)MELDINGEN:

```
INVALID definition window w - window w is verkeerd gedefinieerd
INVALID window              - window nummer niet correct (0<=w<=9)
Illegal function call       - alleen screenmode 0,1,2 toegestaan!
MISSING definition window w - window w is niet gedefinieerd
```

TIPS:

Zorg ervoor dat het betreffende window juist gedefinieerd is. Veranderen van screenmode of width, en zelfs KEYON of KEY OFF kunnen een windowdefinitie ongeldig maken.

gebruiksvoorbeeld:

```
10 CLS:SCREEN 0
20 FOR N=0 TO 23
30 PRINT "DIT IS DE CLS# DEMONSTRATIE DIE DE TEKST UITWIST BINNEN
EEN BEPAALD WINDOW"
60 NEXT N
70 '
80 WINDOW #1,5,5,25,20
90 CLS #1
```

CODE TO DATA *Zet de inhoud van het geheugen om in DATA-regels.*

SYNTAX : CODE TO DATA start,eind[,H]

Start is het startadres en eind is het eindadres van het om te zetten stuk geheugen, waarbij beide adressen tussen 32768 en 65535 dienen te liggen.
Indien optie ',H' aanwezig vindt omzetting plaats in hexadecimale DATA, anders in decimale getallen.

WERKING:

Dit commando geeft u de mogelijkheid om een gedeelte van het geheugen, b.v. een machinetaal routine om te zetten in DATA-regels. De nieuwe regels worden achter het aanwezige programma toegevoegd.
Bij omzetting in decimale getallen worden per regel maximaal 60 data-items gevormd. Bij omzetting in hexadecimale data worden daarvan maximaal 120 op een regel gezet, en wel tussen aanhalingstekens (bijvoorbeeld: 9000 DATA "78AFDC91EEC9").

(FOUT)MELDINGEN:

Out of memory	- het gewenste geheugenblok kan niet in één keer overgezet worden.
Illegal function call	- een opgegeven adres ligt buiten het bereik
MODE error	- dit commando mag alleen direct gegeven worden en niet in een programma voorkomen
LINE number too big	- een regelnummer groter dan 65529 dreigt te ontstaan

TIPS:

De ruimte tussen het einde van het BASIC programma, en de string space wordt als buffer gebruikt. Afhankelijk van de lengte van het BASIC-programma in het geheugen kunnen van enkele bytes tot een paar duizend bytes tegelijk omgezet worden.

gebruiksvoorbeeld:

```
10 SCREEN 0:WIDTH 40
20 PRINT"MOMENT AUB."
30 FOR N=0 TO 250
40 POKE 50000!+N,N
50 NEXT
60 PRINT:PRINT"DRUK <RETURN> EN DAARNA LIST"
70 PRINT:PRINT"CODE TO DATA 50000,50250"
80 LOCATE 0,3
```

COMMON *Beperk het aantal variabelen.*

SYNTAX : COMMON var[,var, ...]

Var, var, ... is een lijst van numerieke, rij- en/of lijstvariabelen, waarbij de variabelen onderling gescheiden dienen te zijn door komma's. Slechts de eerste twee letters van een variabele-naam zijn relevant en dienen dus uniek te zijn; het gebruik van meerdere letters is wel toegestaan.

WERKING:

Na een COMMON-commando blijven alleen de daarachter benoemde variabelen in het geheugen behouden; alle andere worden gewist. Een COMMON-instructie zal vaak gebruikt worden direct voor een CHAIN-commando, om het aantal variabelen te beperken dat meegenomen wordt naar een volgend programma.

AANVERWANTE COMMANDO'S: CHAIN, ERASE

(FOUT)MELDINGEN:

OUT of string space - de "string space" is te klein, gebruik CLEAR

gebruiksvoorbeeld:

```
10 A=50:A$="HALLO"
20 A1=60:A1$="HOI"
30 DIMB(1):DIMB$(1)
40 DIMB1(1):DIMB1$(1)
50 B(0)=5:B(1)=10
60 B1(0)=4:B1(1)=8
70 B$(0)="STRING1":B$(1)="STRING2"
80 B1$(0)="B1$(0)":B1$(1)="B1$(1)"
90 '
100 GOSUB 160
110 PRINT"COMMON A,A$,B(),B$()":PRINT
120 COMMON A,A$,B(),B$()
130 GOSUB 160
140 END
150 '
160 PRINT"A     =";A,
170 PRINT"A1     =";A1
180 PRINT"A$    =";A$,
190 PRINT"A1$    =";A1$
200 PRINT"B(0) =";B(0),
210 PRINT"B1(0) =";B1(0)
220 PRINT"B(1) =";B(1),
230 PRINT"B1(1) =";B1(1)
240 PRINT"B$(0)=";B$(0),
250 PRINT"B1$(0)=";B1$(0)
260 PRINT"B$(1)=";B$(1),
270 PRINT"B1$(1)=";B1$(1)
280 PRINT
290 RETURN
```

COPLIN *Kopieer BASIC-regels.*

SYNTAX : COPLIN beginregel,eindregel,bestemming

Beginregel is de eerste, en eindregel de laatste regel van het BASIC-programma dat gekopieerd dient te worden. Bestemming geeft aan het regelnummer waarachter het gekopieerde stuk dient te worden ingevoegd. Eindregel dient groter te zijn dan beginregel, en bestemming dient hetzij groter te zijn dan eindregel, hetzij kleiner dan beginregel.

WERKING:

Kopieert BASIC-tekstregels naar elders in het programma, waarbij de orginele regels bewaard blijven. De kopie wordt direct achter de opgegeven bestemming geplaatst, met een nummerophoging van 1. Zo nodig wordt in het programma vanaf bestemming de bestaande nummering veranderd om ruimte te maken voor de nieuwe regels. Mochten er regelnummers boven 65529 dreigen te ontstaan, dan vindt automatisch een hernummering van het gehele programma plaats (RENUM 10,10,10).

AANVERWANTE COMMANDO'S: MOVLIN, DELETE, RENUM

(FOUT)MELDINGEN:

Out of memory	- er is te weinig geheugenruimte om het opgegeven aantal regels in één keer te kopieren
COPLIN error	- de opgegeven waarden voor beginregel, eindregel of bestemming zijn niet correct
MODE error	- het commando kan alleen direct ingetypt worden en mag niet in een programma opgenomen zijn
LINE number too big RENUM started	- een te hoog regelnummer is ontstaan; automatische hernummering (10,10,10) vindt plaats
Undefined line number	- een der opgegeven regelnummers bestaat niet

TIPS:

De ruimte tussen het einde van het BASIC programma, en de string-ruimte wordt als tijdelijke opslag gebruikt. Het aantal regels dat in één keer gekopieerd kan worden is daarom afhankelijk van de lengte van programma.

gebruiksvoorbeeld:

```
10 ' TYP: COPLIN 10,30,70 en druk dan <RETURN>
20 ' OM DE REGELS 10 T/M 30
30 ' TE KOPIEREN ACHTER REGEL 70
40 '
50 ' DOE DAARNA LIST OM TE KIJKEN
60 ' HOE HET PROGRAMMA VERANDERD IS
70 LIST:END
```

DEF PROC *Geef het begin van een procedure aan.*

SYNTAX : DEF PROC "naam procedure"

Naam procedure is een rij lettertekens, van maximaal 200 lettertekens, waarbij hoofd- en kleine letters een onderscheiden betekenis hebben. De procedurenaam moet verplicht tussen aanhalingstekens staan. Het commando DEFPROC mag uitsluitend aan het begin van een BASIC-regel gebruikt worden, en er mogen geen statements in dezelfde regel op volgen.

WERKING:

Dit commando geeft aan waar een procedure begint. Elke procedure begint met een DEFPROC"procedurenaam" en eindigt bij de eerstvolgende ENDPROC. Tussen een DEFPROC en een ENDPROC mag een stuk programma staan met alle mogelijke statements. Vermijd echter daarbij het gebruik van sprongopdrachten die uit de procedure kunnen springen omdat dit een correcte terugkeer uit de procedure kan bemoeilijken.
Procedures mogen niet genest worden!

AANVERWANTE COMMANDO'S: PROC, ENDPROC, SAVEPROC, MERGEPROC, DELETEPROC, LISTPROC, LLISTPROC

(FOUT)MELDINGEN:

DEFPROC op zich zal nooit een foutmelding opleveren, maar wel de commando's die de DEFPROC gebruiken: PROC,ENDPROC,SAVEPROC,MERGEPROC en DELETEPROC.

TIPS:

Denk eraan dat verschil wordt gemaakt tussen hoofd- en kleine letters. Wanneer ook gebruik gemaakt wordt van het commando SAVEPROC zal de naam van het op disk/cassette gezette bestand ingekort worden tot 6 of 11 karakters (bij disk ook omgezet in hoofdletters).
Bij het terugladen via MERGEPROC moet dan ook die naam van 6 of 11 letters gebruikt worden. De naam in het DEFPROC-statement zelf blijft echter onveranderd.
Om verwarring hierin te voorkomen, dient men bij voorkeur bij DEFPROC namen te gebruiken van hooguit 6 of 11 karakters.

gebruiksvoorbeeld:

```
10 PROC"INPUT"
20 PROC"OUTPUT"
30 END
100 DEFPROC"INPUT"
110 INPUT"UW NAAM:";N$
120 ENDPROC
200 DEFPROC"OUTPUT"
210 PRINT"HALLO ";N$
220 ENDPROC
```

DELETE PROC *Wis een procedure.*

SYNTAX : DELETE PROC "procedurenaam"

Naam procedure is een rij lettertekens, van maximaal 200 lettertekens, waarbij hoofd- en kleine letters een onderscheiden betekenis hebben.

WERKING:

Wist uit een BASIC-programma het gedeelte vanaf een DEFPROC met de opgegeven procedurenaam tot en met de eerstvolgende ENDPROC.

AANVERWANTE COMMANDO'S: PROC, DEFPROC, ENDPROC, SAVEPROC, MERGEPROC, LISTPROC,LLISTPROC

(FOUT)MELDINGEN:

DEFPROC not found - de opgegeven procedurenaam is niet gevonden
ENDPROC not found - de opgegeven procedure mist zijn ENDPROC.
MODE error - het commando kan uitsluitend als directe opdracht ingetypt worden, niet in een programma

gebruiksvoorbeeld:

```
DELETEPROC "INPUT"
```

DFILES *Toon uitgebreid overzicht diskinhoud.*
LDFILES *Print uitgebreid overzicht diskinhoud.*

SYNTAX : DFILES
LDFILES

WERKING:

Geeft net als het normale FILES-commando een overzicht van de files op een diskette, maar geeft daarbij tevens aan hoe lang deze zijn, hoeveel er zijn, en hoeveel vrije ruimte op de disk overblijft.

AANVERWANTE COMMANDO'S: FILES, TFILES, LTFILES, SETDRIVE

(FOUT)MELDINGEN:

Disk offline - geen drive aangesloten, of geen diskette in drive

TIPS:

Dit commando geeft alleen een overzicht van de bestanden op de actuele drive. Een andere drive kan desgewenst geselecteerd worden via het nieuwe SETDRIVE-commando.
Het is niet mogelijk achter DFILES een selectie aan te geven van bestanden, zoals dat wel mogelijk is met FILES.

gebruiksvoorbeeld:

DFILES

DPEEK *Geef de inhoud van 2 geheugenadressen te zamen.*

SYNTAX : DPEEK(adres)

Adres is de eerste van 2 opeenvolgende geheugenadressen, waarbij dient te gelden: -32768 <= adres <= 65535.
DPEEK is een functie, en kan dus bijvoorbeeld gebruikt worden in:

```
PRINT DPEEK(50000)
A=DPEEK(60000)
IF DPEEK (63000)=512 THEN .....
```

WERKING:

Deze functie berekent de inhoud van een geheugenlocatie, en telt er 256* de inhoud van de volgende geheugenlocatie bij op. Op deze manier kan snel een waarde worden uitgelezen die in 2 opvolgende bytes ligt opgeslagen.
Indien een negatief getal wordt opgegeven, wordt hiervan een positief getal gemaakt door het af te trekken van 65536.

AANVERWANTE COMMANDO'S: PEEK, POKE, DPOKE

(FOUT)MELDINGEN:

Overflow - het opgegeven adres ligt buiten de mogelijke waarden

gebruiksvoorbeeld:

```
10 PRINT "HET HOOGSTE GEHEUGENADRES DAT BASIC"
20 PRINT "MOMENTEEL KAN GEBRUIKEN IS:";
30 PRINT DPEEK(&H FC4A)
```

DPOKE *zet een getal in 2 opeenvolgende geheugenadressen.*

SYNTAX : DPOKE adres,inhoud

Adres is de eerste van 2 opeenvolgende geheugenadressen, waarbij dient te gelden: -32768 <= adres <0 of 32768<= adres <=65535.
Inhoud is het getal dat in de twee opeenvolgende geheugenplaatsen gezet wordt, waarbij geldt: 0 <= inhoud <= 65535.
Indien een negatief getal wordt opgegeven, wordt dit omgezet in een positief getal door het af te trekken van 65536.

WERKING:

Deze instructie voert als het ware een dubbele POKE uit, door de gewenste inhoud te delen door 256 en dit in het opgegeven adres te zetten terwijl het restant in het daaropvolgende geheugenadres gezet wordt.

AANVERWANTE COMMANDO'S: DPEEK, PEEK, POKE

(FOUT)MELDINGEN:

Overflow	- het opgegeven adres en/of de aangegeven inhoud ligt buiten de mogelijke waarden
Illegal function call	- het opgegeven getal ligt tussen 0 en 32767; in de ROM valt niet te POKEn

gebruiksvoorbeeld:

```
10 FOR I = 60000 TO 60010 STEP 2
20 DPOKE I,1000
30 NEXT I
```

END PROC *Geef het eind van een procedure-definitie aan.*

SYNTAX : END PROC

WERKING:

**Na een PROC-commando wordt een procedure vanaf de aangeroepen DEFPROC tot aan de eerstvolgende ENDPROC uitgevoerd.
Wanneer tijdens de uitvoering van een BASIC-programma een DEFPROC ontmoet wordt, zonder dat deze met een PROC is aangeroepen, gaat het programma verder na de eerstvolgende ENDPROC.**

AANVERWANTE COMMANDO'S: PROC, DEFPROC, DELETEPROC, SAVEPROC MERGEPROC, LISTPROC, LLISTPROC

(FOUT)MELDINGEN:

MODE error - **het commando kan uitsluitend in een programma opgenomen worden.**

gebruiksvoorbeeld:

```
10 PROC"INPUT"
20 PROC"OUTPUT"
30 END
40 '
50 DEFPROC"INPUT"
60 INPUT"UW NAAM:";N$
70 ENDPROC
80 '
90 DEFPROC"OUTPUT"
100 IF POL THEN LPRINT"HALLO ";N$
110 PRINT"HALLO ";N$
120 ENDPROC
```

FILL # *Vul de inhoud van een window met een karakter.*

SYNTAX : FILL #window,waarde

Window is het nummer van een tevoren gedefinieerd window, en waarde is de ASCII-waarde van het gewenste karakter (tussen 0 en 255).

WERKING:

Vult het scherm binnen het opgeven window met het karakter waarvan de ASCII-waarde opgegeven is.
Dit commando werkt alleen in SCREEN 0 en 1, niet in SCREEN 2 !

AANVERWANTE COMMANDO'S: CLS#, WINDOW#

(FOUT)MELDINGEN:

```
INVALID definition window w - window w is verkeerd gedefinieerd
INVALID window              - window nummer niet tussen 0 en 9
Illegal function call       - alleen in SCREEN 0 en 1 toegestaan!
MISSING definition window w - window w is niet gedefinieerd
```

TIPS:

Zorg ervoor dat het betreffende window juist gedefinieerd is. Veranderen van screenmode of width, en zelfs KEYON of KEY OFF kunnen een window definitie ongeldig maken.
Dit commando werkt niet in SCREEN 2.

gebruiksvoorbeeld:

```
10 CLS:SCREEN 0
20 WINDOW #1,5,5,25,20
30 '
40 FOR N=0 TO 255
50  FILL #1,N
60 NEXT N
```

FIND *Zoek eerste voorkomen tekst in BASIC-programma*
FIND NEXT *Zoek volgende voorkomen*
FIND ALL *Zoek elk voorkomen*
LFIND *Print eerste voorkomen*
LFIND NEXT *Print volgende voorkomen*
LFIND ALL *Print elk voorkomen*

SYNTAX : FIND
FIND NEXT
FIND ALL
LFIND
LFIND NEXT
LFIND ALL

WERKING:

Deze commando's, die rechtstreeks ingetypt dienen te worden, geven u de mogelijkheid om BASIC keywords, tekst tussen aanhalingstekens, maar ook variabelen-namen in de hele BASIC tekst te zoeken.
Een regel waarin de te vinden tekst voorkomt, wordt op het scherm getoond dan wel uitgeprint.

Na het intypen van een dezer commando's komt de vraag "FINDSTRING? op het scherm, waarna u de te zoeken tekst kunt opgeven.
Wilt u tekst niet alleen zoeken, maar ook tegelijk vervangen, gebruik dan het nieuwe commando REPLACE.

AANVERWANTE COMMANDO'S: REPLACE, LREPLACE

(FOUT)MELDINGEN:

MODE error - de commando's kunnen alleen direct ingetypt worden en niet in een programma voorkomen
Illegal function call - een commando '(L)FINDNEXT wordt gegeven, terwijl de vorige zoekpoging niets opleverde

TIPS:

Een BASIC-statement wordt in het geheugen van de computer op een verkorte wijze bewaard ('tokenized'). Indien de opgegeven te zoeken tekst uit zo'n statement bestaat, of die bevat, dan vindt ook zo'n verkorting plaats alvorens te gaan zoeken.
In dat geval zal alleen een zelfde statement gevonden worden, maar niet hetzelfde woord wanneer dat ergens als een stuk tekst tussen aanhalingstekens staat.
Als u bijvoorbeeld zoekt naar PRINT, wordt wel gevonden
```
10 PRINT A$
```
maar niet
```
20 B$="PRINT DIT UIT!".
```
Als u andersom als te zoeken opgeeft "PRINT" (met aanhalingstekens!), dan wordt alleen letterlijke tekst gevonden, en niet het statement.

GET *Wacht op toetsdruk en slaat toets op in string.*

SYNTAX : GET string

String is een rijvariabele of lijstvariabele.

WERKING:

Wacht totdat er een toets gedrukt wordt, en slaat het karakter dat bij de gedrukte toets hoort op in de opgegeven string. Deze string heeft daarna altijd de lengte 1.

AANVERWANTE COMMANDO'S: PAUSE, INKEY$, INPUT$

(FOUT)MELDINGEN:

Syntax error - de opgegeven variabele is geen rijvariabele (string)

TIPS:
Een enkele toetsindruk kan ook in een string opgeslagen worden, door bijvoorbeeld te gebruiken: A$=INPUT$(1).
Het verschil is, dat GET zorgt dat de keybuffer leeg is, voordat een toetsdruk afgewacht wordt; eventueel eerder ingedrukte toetsen hebben dus geen (onverwacht) effect.

gebruiksvoorbeeld:

```
10 PRINT"10 GET A$"
20 PRINT:PRINT"DIT IS HETZELFDE ALS:":PRINT
30 PRINT"10 A$=INKEY$:IFA$=";STRING$(2,34);"THEN10"
40 '
50 PRINT:PRINT"DRUK EEN TOETS":GET A$
60 IF ASC(A$) <32 THEN A$="EEN EDITING TOETS"
70 PRINT:PRINT"U DRUKTE OP ";A$
80 GOTO 50
```

INIFNK *Initieer functietoetsen.*

SYNTAX: INIFNK

WERKING:

De functietoetsen F1 tot en met F10 worden weer ingevuld, zoals op het moment van het aanzetten van de computer.

gebruiksvoorbeeld:

```
10 SCREEN 0:KEY ON
20 FOR N=1 TO 5
30 KEY N,"KEY "+CHR$(N+48)
40 NEXT
50 '
60 PRINT"DE FUNCTIE TOETSEN ZIJN GEHERDEFINIEERD"
70 PRINT"DRUK EEN TOETS OM ZE TE HERZETTEN ....."
80 '
100 PAUSE 0
110 INIFNK
```

INPSG *Initieer soundgenerator.*

SYNTAX: INIPSG

WERKING:

De programmeerbare geluidsgenerator van de MSX wordt compleet gereset. Indien de computer nog bezig was met geluid, zal dit onmiddellijk stopgezet worden.

gebruiksvoorbeeld:

```
10 RESTORE:SCREEN 0
20 FOR N=6 TO 13
30 READ M
40 PRINT"SOUND";N;",";M
50 SOUND N,M
60 NEXT N
70 '
80 PRINT:PAUSE 2:PRINT"INIPSG":INIPSG
90 END
100 '
110 DATA 31,7,16,16,16,71,2,14
```

KILL BUF *Leeg toetsenbord-buffer.*

SYNTAX: KILL BUF

WERKING:

De MSX heeft een geheugenbuffer, waarin de code van een ingedrukte toets tijdelijk opgeslagen wordt. Ook als een programma niet om een INPUT vraagt, wordt toch een eventueel gedrukte toets opgeslagen.
Zo kan het per ongeluk of te lang indrukken van een toets soms onverwachte effecten hebben in een later deel van een programma.
Dit kan voorkomen worden door, voordat het programma echt een toetsdruk nodig heeft, eerst te garanderen dat de toetsnbord-buffer leeg is. Het commando KILLBUF draagt daarvoor zorg.

gebruiksvoorbeeld:

```
.
.
200 PRINT"< druk op toets van keuze >"
210 KILLBUF
220 I$=INKEY$
230 IF I$="1" THEN CLS:GOTO 1000
.
.
```

KILL REM *Verwijder REM-statements.*
LKILL REM *Verwijder REM-statements met geprint overzicht.*

SYNTAX: KILL REM [,I]
LKILL REM [,I]

WERKING:

REM-statements in een programma zijn van belang voor de inzichtelijkheid van een listing. Ze nemen echter geheugenruimte in beslag, die soms voor andere doeleinden hard nodig is.
KILLREM verwijdert alle REM-statements, waarbij deze desgewenst tegelijk uitgeprint kunnen worden.
De optie ',I' kan achter het statement worden toegevoegd, indien men een intelligente KILLREM wenst. Daarbij wordt voorkomen dat regels verwijderd kunnen worden die uitsluitend uit een REM-statement bestaan, maar waarheen wel een sprong elders vanuit het programma kan plaatsvinden via een GOTO of GOSUB. Deze optie neemt wel meer tijd in beslag.

(FOUT)MELDINGEN:

Mode error - het commando kan alleen direct gegeven worden, en mag niet in een programma voorkomen

TIPS:

Het '-teken is een bijzondere vorm van het REM-statement. Dit teken kan niet zonder meer door een KILLREM-opdracht gevonden en verwijderd worden.
Gebruik in dergelijke gevallen eerst de opdracht REPLACEALL, antwoord op de vraag FINDSTRING? met ' en antwoord op de vraag REPLACE BY? met REM.
Daarna zal een REMKILL alle REM's verwijderen.

gebruiksvoorbeeld:

```
KILLREM,I
```

LIST DATA *Toon regels met DATA-statements.*
LLIST DATA *Print regels met DATA-statements.*

SYNTAX: LIST DATA
LLIST DATA

WERKING:

Geeft een overzicht van alle BASIC-regels van een programma die DATA-statements bevatten, naar keuze op het beeldscherm of de printer.

AANVERWANTE COMMANDO'S: DATA, (L)LIST, (L)LISTGO, (L)LISTLIN, (L)LISTPROC, (L)LISTVAR, (L)LISTWINDOW

LIST GO *Toon BASIC-regels met sprongopdrachten.*
LLIST GO *Print BASIC-regels met sprongopdrachten.*

SYNTAX: LIST GO
LLIST GO

WERKING:

Geeft een overzicht van alle BASIC-regels van een programma, waarin sprongopdrachten voorkomen (b.v. GOTO 100, GOSUB 9000, ... THEN 900 ELSE 950).
Achter het getoonde regelnummer verschijnen de regelnummers met het doel van de sprong.

AANVERWANTE COMMANDO'S: GOTO, GOSUB, THEN, ELSE, RETURN, RESUME
(L)LIST (L)LISTDATA,(L)LISTLIN,
(L)LISTPROC, (L)LISTVAR, (L)LISTWINDOW

LIST LIN *Toon bestemmingsregels van sprongopdrachten.*
LLIST LIN *Print bestemmingsregels van sprongopdrachten.*

SYNTAX: LIST LIN
LLIST LIN

WERKING:

Geeft een overzicht van alle BASIC-regels, waarheen gesprongen kan worden vanaf elders in een programma.
Achter het getoonde regelnummer verschijnen de regelnummers waarin de betreffende sprongopdrachten voorkomen.

AANVERWANTE COMMANDO'S: GOTO, GOSUB, THEN, ELSE, RETURN, RESUME, (L)LIST, (L)LISTDATA, (L)LISTGO, (L)LISTPROC (L)LISTVAR, (L)LISTWINDOW

LIST PROC *Toon regels die een DEFPROC bevatten.*
LLIST PROC *Print regels die een DEFPROC bevatten.*

SYNTAX: LIST PROC
LLIST PROC

WERKING:

Geeft een overzicht van alle BASIC regels die een DEFPROC bevatten.
De betreffende regels worden geheel afgebeeld.

AANVERWANTE COMMANDO'S: DEFPROC, (L)LIST, (L)LISTDATA, (L)LISTGO, (L)LISTLIN, (L)LISTVAR, (L)LISTWINDOW

LIST PSG *Toon huidige waardən van de geluidsgenerator.*
LLIST PSG *Print huidige waarden van de geluidsgenerator.*

SYNTAX: LIST PSG
LLIST PSG

WERKING:

Geeft een overzicht van de op dat moment geldende waarden van alle registers van de programmeerbare geluidsgenerator (PSG).
De waarden van de registers 7, 14 en 15 worden in een binaire vorm getoond.

AANVERWANTE COMMANDO'S: SOUND, PLAY, INIPSG

LIST SCREEN *Toon huidige SCREEN-waarden.*
LLIST SCREEN *Print huidige SCREEN-waarden.*

SYNTAX: LIST SCREEN
LLIST SCREEN

WERKING:

Geeft een overzicht van de op dat moment geldende waarden, die met een SCREEN-commando kunnen worden ingesteld:
Schermmode, sprite grootte, toetsklik, baudrate,en printertype.

Geeft bij baudrate tevens aan of een afwijkende snelheid is geselecteerd middels het nieuwe BAUD-commando. LISTSCREEN hanteert daarbij de volgende aanduidingen:

Baudrate	aanduiding
900	A
1200	1
1500	C
1800	D
2100	E
2400	2
2700	G
3000	H

AANVERWANTE COMMANDO'S: SCREEN, BAUD

LIST TYPE *Toon type-definities variabelen.*
LLIST TYPE *Print type-definities variabelen.*

SYNTAX: LIST TYPE
LLIST TYPE

WERKING:

Bij het aanzetten van de computer worden alle variabelen standaard ingesteld op het type DBL (dubbele precisie).
Desgewenst kan de programmeur dit veranderen met een der commando's DEFINT, DEFSNG, DEFDBL of DEFSTR.
LISTTYP laat van alle beginletters van variabelen zien hoe ze gedefinieerd zijn, door achter het karakter een der volgende aanduidingen te geven: % INT (geheel getal), ! SNG (enkele precisie), # DBL (dubbele precisie) of $ STR (rijvariabele, string).

AANVERWANTE COMMANDO'S: DEFINT, DEFSNG, DEFDBL, DEFSTR

LIST USR *Toon waarden van USR-functies.*
LLIST USR *Print waarden van USR-functies.*

SYNTAX: LIST USR
LLIST USR

WERKING:

Geeft een overzicht van alle 10 adressen die door een USR-functie zullen worden aangeroepen. Als een USR-functie niet door de gebruiker is gedefinieerd via een DEFUSR-commando, dan zal het adres 18266 zijn. Dit is het adres in de MSX-ROM waar de foutmelding 'Illegal function call' uitgevoerd wordt.

AANVERWANTE COMMANDO'S: DEFUSR, USR

LIST VAR *Toon gebruik variabelen in programma.*
LLIST VAR *Print gebruik variabelen in programma.*

SYNTAX: LIST VAR
LLIST VAR

WERKING:

Geeft een alfabetisch overzicht van alle in een BASIC-programma gebruikte variabelen, alsmede de regels waarin deze variabelen worden gebruikt. Ondanks het feit dat een van de snelste sorteermethoden gebruikt wordt, kan het bij een lang programma even duren voor de lijst verschijnt.

AANVERWANTE COMMANDO'S: (L)LIST, (L)LISTDATA, (L)LISTGO
(L)LISTLIN, (L)LISTPROC, (L)LISTWINDOW

LIST WINDOW *Toon huidige definities windows.*
LLIST WINDOW *Print huidige definities windows.*

SYNTAX: LIST WINDOW
LLIST WINDOW

WERKING:

Geeft een overzicht van de definities van alle windows die op dat moment gelden.

AANVERWANTE COMMANDO'S: (L)LIST, (L)LISTDATA, (L)LISTGO
(L)LISTLIN, (L)LISTPROC, (L)LISTVAR

LOCATE # *Positioneer cursor in window.*

SYNTAX: LOCATE #window,x-pos,y-pos

Window is het gewenste window-nummer (tussen 0 en 9), x-pos is de relatieve x-positie en y-pos de relatieve y-positie binnen het window.

WERKING:

Plaats de cursor op de opgegeven plaats in het window. Om de fysieke positie van de cursor te bepalen moeten de linksboven-coördinaten van het bijbehorende window erbij opgeteld worden.
De cursor binnen een window blijft onzichtbaar, totdat een WPRINT of WINPUT-statement gebruikt wordt. Van elke window afzonderlijk wordt een cursorpositie bijgehouden.

AANVERWANTE COMMANDO'S: WPRINT #, WINPUT #

(FOUT)MELDINGEN:

LOCATE out of window w	- de coördinaten liggen buiten het window
INVALID window	- windownummer niet tussen 0 en 9
INVALID definition window w	- window w is verkeerd gedefinieerd
MISSING definition window w	- window w is niet gedefinieerd
Illegal function call	- alleen screenmodes 0,1,2 toegestaan

TIPS:

Zorg ervoor dat het betreffende window juist is gedefinieerd.
Veranderen van screenmode of width, en zelfs KEYON of KEYOFF kunnen een window definitie ongeldig maken!

gebruiksvoorbeeld:

```
10 SCREEN 1
20 WINDOW #1,5,5,10,10
30 WINDOW #2,5,11,10,17
40 LOCATE #1,2,2:WPRINT #1;"A"
50 LOCATE #2,3,0:WPRINT #2;"B"
```

LOWER *zet hoofdletters om in kleine letters.*

SYNTAX: LOWER string

String is een rijvariabele of lijstvariabele.

WERKING:

Dit commando verandert voorkomende hoofdletters in een string in kleine letters.

AANVERWANTE COMMANDO'S: UPPER, CAPSON, CAPSOFF

TIPS:

LOWER is een commando, en geen functie. Een expressie als A$=LOWER B$ is dus niet mogelijk!
De omzetting vindt plaats binnen de aangegeven string.

gebruiksvoorbeeld:

```
10 SCREEN 0
20 CAPS ON
30 INPUT"TYP EEN WOORD EN DRUK <RETURN>";W$
40 PRINT"U TYPTE: ";W$
50 LOWER W$
60 PRINT"DE FUNKTIE 'LOWER' MAAKT HIERVAN: ";W$
```

MERGE PROC *Koppel een procedure aan programma vast.*

SYNTAX: MERGE PROC "[dev:]procnaam"

Dev is de aanduiding voor het opslagmedium (CAS, A-H of MEM), procnaam is een rij lettertekens met een maximum van 6 (cassetterecorder) of 11 (bij diskdrive); dev mag eventueel weggelaten worden (vergelijk de regels bij de normale LOAD-opdrachten).

WERKING:

De procedure wordt achter het huidige programma geplaatst, met regelnummers oplopend met 1 vanaf het oude laatste regelnummer.
De te MERGEn procedure behoeft niet als ASCII file gesaved te zijn.

AANVERWANTE COMMANDO'S: SAVEPROC, MERGE

(FOUT)MELDINGEN:

File not found	- de te laden procedure staat niet op deze disk
LINE number too big	- tijdens het MERGEn is een regelnummer groter dan 65529 ontstaan; het MERGEn is afgebroken

TIPS:

Dit commando kan ook gebruikt worden om BASIC-programma's te koppelen, waarbij in tegenstelling tot bij het normale MERGE-commando hier niet de eis geldt dat zo'n programma als ASCII gesaved dient te zijn.
Wel dienen in het te mergen programma geen GOTO's en GOSUB's voor te komen; bij het geven van nieuwe regelnummers worden deze namelijk niet opnieuw genummerd!

gebruiksvoorbeeld:

```
MERGEPROC "B:GELUID3.NSE"
```

MOVLIN *Verplaats BASIC-regels*

SYNTAX: MOVLIN beginregel,eindregel,bestemming.

Beginregel is de eerste, en eindregel de laatste regel van het stuk BASIC-programma dat verplaatst dient te worden. Bestemming geeft aan het regelnummer waarachter het verplaatste stuk wordt ingevoegd. Eindregel dient groter te zijn dan beginregel, en bestemming dient hetzij groter te zijn dan eindregel, hetzij kleiner dan beginregel.

WERKING:

Verplaatst BASIC tekstregels naar elders in het programma. De regels worden oplopend met 1 genummerd vanaf de bestemming. Zodra een regelnummer gecreëerd moet worden dat al bestaat, wordt met het verplaatsen gestopt (MOVLIN error). Alle tekstregels die GOTO's en GOSUB'S bevatten die verwijzen naar de te verplaatsen regels worden automatisch opnieuw genummerd.

AANVERWANTE COMMANDO'S: COPLIN, RENUM

(FOUT)MELDINGEN:

MOVLIN error	- de opgegeven waarden voor beginregel, eindregel of bestemming zijn niet correct; er dreigt een regelnummer te worden gemaakt dat al bestaat; tijdens het verplaatsen dreigt een regelnummer groter dan 65529 te ontstaan
MODE error	- dit commando is alleen als directe opdracht mogelijk en mag niet in een programma voorkomen
Undefined line number	- een van de opgegeven regelnummers bestaat niet

TIPS:

Zorg er altijd voor dat er voldoende ruimte is tussen de regelnummers waarheen het gewenste blok verplaatst wordt.
Gebruik voor het verplaatsen van een blok naar het begin van het programma eerst een hulpregel [bijvoorbeeld 0 REM] en wis deze weer na het verplaatsen.

gebruiksvoorbeeld:

```
MOVLIN 200,290,3000
```

PAUSE *Wacht.*

SYNTAX: PAUSE tijd

Tijd is het aantal seconden wachttijd (maximum is 65535).
Indien als tijd 0 (nul) wordt ingevuld, wordt gewacht tot de eerstvolgende toetsdruk.

WERKING:

De uitvoering van het programma wordt het opgegeven aantal seconden stilgezet. Wanneer als tijd 0(nul) is opgegeven, wordt de toetsenbordbuffer leeggemaakt en gewacht op de eerstvolgende toetsdruk.
Gedurende de werking van een PAUSE-commando wordt een eventuele telling middels een ON INTERVAL - opdracht stopgezet.
Tussentijds afbreken van PAUSE via CTRL-STOP is mogelijk.

AANVERWANTE COMMANDO'S: STOP, FOR...NEXT, GET

gebruiksvoorbeeld:

```
.
.
900 LOCATE 0,19:PRINT" druk een toets ...."
910 PAUSE 0
920 CLS
.
.
```

POL *Check of printer aanstaat.*

SYNTAX: POL

WERKING:

POL is een functie, die de waarde 255 oplevert als een printer aangesloten is en on-line staat; zo niet dan levert het de waarde 0 (nul). Deze functie kan een tekortkoming in de MSX-ROM opvangen; indien geen printer aan staat, zal een LPRINT-statement tot in het oneindige blijven wachten tot iemand op het idee komt de printer wel aan te zetten. Met de POL-functie kan nu een waarschuwing gecreëerd worden, of desnoods een sprong gemaakt worden tot voorbij LPRINT-statements.

gebruiksvoorbeeld:

```
.
.
350 IF POL THEN 380
360 BEEP:LOCATE 0,19:PRINT"Zet printer aan !!!"
370 PAUSE 5:GOTO 350
380 LPRINT A$
.
.
```

PROC *Roep een procedure aan.*

SYNTAX: PROC"procedurenaam"

Procedurenaam is een rij lettertekens (maximaal 200) waarbij hoofd- en kleine letters een onderscheiden betekenis hebben.

WERKING:

De uitvoering van het BASIC-programma gaat verder vanaf de DEFPROC met de opgegeven procedurenaam tot en met de eerstvolgende ENDPROC. Daarna keert de uitvoering terug naar het statement volgend op de PROC.

AANVERWANTE COMMANDO'S: DEFPROC, ENDPROC, SAVEPROC, MERGEPROC, DELETEPROC, LISTPROC, LLISTPROC

(FOUT)MELDINGEN:

DEFPROC not found - een procedure met de opgegeven naam komt niet voor
ENDPROC not found - de aangeroepen procedure wordt nergens afgesloten

TIPS:

Denk eraan dat verschil gemaakt wordt tussen HOOFD en kleine letters in de procedurenaam.
Procedures kunnen desgewenst genest worden, dus een andere procedure mag aangeroepen worden tussen een DEFPROC en een ENDPROC.

gebruiksvoorbeeld:

```
10 PROC "INPUT"
20 PROC "OUTPUT"
30 END
40 '
50 DEFPROC "INPUT"
60 INPUT"UW NAAM:";N$
70 ENDPROC
80 '
90 DEFPROC "OUTPUT"
100 IF POL THEN LPRINT"HALLO ";N$
110 PRINT"HALLO ";N$
120 ENDPROC
```

QUIT *Verlaat DELTA BASIC.*

SYNTAX: QUIT

WERKING:

Met dit commando kunt u de DELTA BASIC verlaten. Er zal om een bevestiging gevraagd worden: "ARE YOU SURE?". Druk op de Y-toets als u DELTA BASIC daadwerkelijk wilt verlaten.

Door op de RESET-toets te drukken, zal DELTA BASIC niet uit het geheugen verdwijnen. Het enige andere alternatief is het uitzetten van de computer.

REPLACE *Vervang eerste voorkomen tekst in BASIC-programma.*
REPLACE NEXT *Vervang volgend voorkomen.*
REPLACE ALL *Vervang elk voorkomen.*
LREPLACE *Print en vervang eerste voorkomen.*
LREPLACE NEXT *Print en vervang volgend voorkomen.*
LREPLACE ALL *Print en vervang elk voorkomen.*

SYNTAX: REPLACE
REPLACE NEXT
REPLACE ALL
LPREPLACE
LREPLACE NEXT
LREPLACE ALL

WERKING:

Deze commando's die rechtstreeks ingetypt dienen te worden, geven u de mogelijkheid om BASIC-keywords, tekst tussen aanhalingstekens, maar ook variabelen-namen in de hele BASIC-tekst te vervangen.
Een regel waarin de te vervangen tekst voorkomt, wordt op het scherm getoond dan wel uitgeprint.

Na het intypen van een dezer commando's komt de vraag "FINDSTRING?", waarna u de te vervangen tekst kunt opgeven. Na de vraag "REPLACE BY?" geeft u aan door welke tekst een en ander vervangen moet worden.

Wilt u een tekst alleen zoeken, doch niet vervangen, gebruik dan het nieuwe commando FIND.

AANVERWANTE COMMANDO'S: FIND [NEXT,ALL], LFIND [NEXT,ALL]

(FOUT)MELDINGEN:

MODE error	- de commando's mogen alleen direct ingetypt worden en niet in een programma voorkomen
Illegal function call	- een commando (L)REPLACE NEXT wordt gegeven, terwijl de vorige vervangpoging geen resultaat had

TIPS:

Na een commando REPLACE mag een commando FIND NEXT volgen; andersom kan een commando FIND echter niet gevolgd worden door REPLACE NEXT.
Zie verder ook de tips bij het nieuwe commando FIND.

ROLLD # *Rol inhoud window omlaag.*
ROLLL # *Rol inhoud window naar links.*
ROLLR # *Rol inhoud window naar rechts.*
ROLLU # *Rol inhoud window omhoog.*

SYNTAX: ROLLD #window[,aantal]
ROLLL #window[,aantal]
ROLLR #window[,aantal]
ROLLU #window[,aantal]

Window is het nummer tussen 0 en 9 van een tevoren gedefinieerd window. Windownummer 0 (nul) laat het gehele scherm rollen.
Aantal is het aantal keren dat direct achter elkaar gerold dient te worden; aantal is een geheel getal tussen 1 en 255. Aantal is een optie en mag ook weggelaten worden (hetzelfde als aantal =1).

WERKING:

Rolt de inhoud van een window (tekst of grafiek) het opgegeven aantal posities in de gewenste richting. Daarbij valt in de opgegeven richting steeds een regel of kolom buiten het window en verdwijnt; aan de tegenovergestelde zijde wordt een lege regel of kolom toegevoegd.
Een positie schuiven in SCREEN 2 betekent 8 pixels tegelijk.

AANVERWANTE COMMANDO'S: WRAPL #, WRAPR #, WRAPU #, WRAPD #, WINDOW #

(FOUT)MELDINGEN:

INVALID definition window w	- indow w is verkeerd gedefinieerd
INVALID window	- window nummer niet correct (0<=w<=9)
Illegal function call	- alleen screenmode 0,1,2 toegestaan!
MISSING definition window w	- window w is niet gedefinieerd

TIPS:

Zorg ervoor dat het betreffende window juist gedefinieerd is. Veranderingen van screenmode of width, en zelfs KEYON of KEYOFF kunnen een window definitie ongeldig maken.

gebruiksvoorbeeld:

```
10 CLS:SCREEN 0:KEY OFF:WIDTH 40
20 WINDOW #1,1,1,38,10
30 WINDOW #2,1,13,18,20
40 WINDOW #3,23,13,38,20
50 FOR W=1 TO 3: FOR N=1 TO 30
60 WPRINT #W;"DIT IS WINDOW";W;
70 NEXT N:NEXT W
80 ROLLU #1,4
90 ROLLL #2,10
100 ROLLR #3,10
110 ROLLD #1,10
120 ROLLD #0,24
```

SAVE PROC *Bewaar een procedure op tape of disk.*

SYNTAX: SAVE PROC"[dev:]procnaam" [,A]

Dev is de aanduiding voor het opslagmedium (CAS, A-H of MEM), procnaam is een rij lettertekens met een maximum van 6 (cassetterecorder) of 11 (bij diskdrive); dev: mag eventueel weggelaten worden (vergelijk de regels bij de normale SAVE-opdrachten).
Indien achter de opdracht de optie ',A' toegevoegd wordt, wordt de procedure als ASCII weggeschreven, als deze optie weggelaten is, vindt het wegschrijven 'tokenised' plaats.

WERKING:

Zoekt de DEFPROC die bij de opgegeven procnaam behoort, en schrijft het stuk BASIC vanaf deze regel tot de eerstvolgende ENDPROC weg.
De procedure kan tokenised, maar ook als ASCII file weggeschreven worden. Voor een eventuele MERGEPROC maakt dit geen verschil.

AANVERWANTE COMMANDO'S: MERGEPROC, SAVE, CSAVE

(FOUT)MELDINGEN:

DEFPROC not found - de opgegeven procedurenaam is niet gevonden
ENDPROC not found - de opgegeven procedure mist een ENDPROC

TIPS:

Zorg ervoor dat de opgegeven procedurenaam exact overeenkomt met de naam die achter de bedoelde DEFPROC staat. Merk daarbij op dat een SAVEROC zelfs een naam van 200 tekens aanvaardt (net als DEFPROC), doch bij het saven worden van die naam slechts 8 of 11 letters overgenomen ! Zowel bij DEFPROC als bij SAVEPROC wordt onderscheid gemaakt tussen hoofd- en kleine letters, doch zodra een bestand op disk staat zal de naam uitsluitend uit hoofdletters bestaan (op cassette blijft het onderscheid wel)!

SCREEN DUMP *Druk de inhoud van het scherm af op printer.*

SYNTAX: SCREEN DUMP

Behalve door het geven van het commando, kan een screendump ook verkregen worden door het tegelijk drukken van de SELECT- en ESC-toetsen.

WERKING:

Het commando kan een screendump maken van SCREEN 0, 1 of 2.
Kleuren worden daarbij in zwart-wit configuratie weergegeven.
Het printen van een screen geschiedt in zogenaamde bit-image mode. Derhalve dient dit vooraf ingesteld te worden via het nieuwe BISET-commando (zie aldaar). Zeker bij een screendump van SCREEN 2 kan het wenselijk zijn in het BISET-commando extra codes op te nemen om de verticale regelinstelling van de printer te verkleinen. Desgewenst kan dit ook gebeuren door middel van een aantal LPRINT-commando's.
Een screemdump kan afgebroken worden door CTRL-STOP te drukken.

AANVERWANTE COMMANDO'S: BIPRINT, BISET

TIPS:

De instellingen van eventueel tevoren gebruikte commando's BISET en BIPRINT worden bij het maken van een screendump gebruikt. Dit geldt echter niet voor de bij BIPRINT ingestelde pixelbreedte; deze wordt automatisch op 6 of 8 gezet, afhankelijk van de screenmode.

Indien in een screendump horizontale witte strepen ontstaan, dient de verticale regelafstand verkleind te worden; een afstand van 1/8 of 1/9" geeft meestal goede resultaten; de codes hiervoor kunt u in het BISET-commando 'meenemen'; vergeet niet voor normaal gebruik de regelafstand weer terug te zetten (meestal 1/6").
Indien bij een screendump alle 24 'regels' afzonderlijk op de kop worden afgedrukt, dient u eerst het commando BIPRINT(8,U) te gebruiken.
Zie ook Bijlage A voor een juist gebruik van het BISET-commando.

Het screendump-commando kan niet altijd gebruikt worden voor het afdrukken van schermen uit commerciële programma's; diverse programma's zullen namelijk Delta Basic overschrijven in het geheugen (zie ook pagina 8 van deze handleiding: geheugengebruik).

SCREEN ON *Schakel het beeldscherm aan.*
SCREEN OFF *Schakel het beeldscherm uit.*

SYNTAX: SCREEN ON
SCREEN OFF

WERKING:

Schakelt het beeldscherm aan of uit. Als het scherm uitgeschakeld is, kan er normaal op geprint of getekend worden, maar een en ander zal pas zichtbaar zijn na het commando SCREEN ON.

AANVERWANTE COMMANDO'S: SCREENSAVE

TIPS:

Denk er aan dat het lijkt of uw computer niets meer doet als het scherm uitstaat. Houd hiermee rekening door b.v. een funktietoets te definieren als KEYn,"SCREENON"+CHR$(13) of door een ON STOP GOSUB constructie toe te passen.

gebruiksvoorbeeld:

```
10 SCREEN 0
20 SCREEN OFF
30 LOCATE 10,10:PRINT"GEHEIM"
40 GET A$
50 IF A$<>"@" THEN 40
60 SCREEN ON
```

SCREEN SAVE *Schakel het beeldscherm na bepaalde tijd uit.*

SYNTAX: SCREEN SAVE [tijd]
SCREEN SAVE OFF

Tijd is de wachttijd in seconden (tussen 1 en 1200).(default-waarde). Voor tijd kan ook 0 (nul) opgegeven worden; dit heeft hetzelfde effect als SCREENSAVE OFF.

WERKING:

Dit commando kan nuttig zijn om de monitor te sparen (inbranden!). Als gedurende de ingestelde tijd geen toets wordt ingedrukt, dan zal het beeldscherm uitgeschakeld worden. Een willekeurige toetsdruk zet het scherm weer aan.
Dit commando werkt uitsluitend zolang geen programma gerund wordt. De uitvoering van een programma zal het commando onderdrukken; 150 seconden na beeindiging van een programma wordt het scherm weer leeggemaakt. Na het inladen van DELTA BASIC is de SCREENSAVE automatisch van toepassing (wachttijd 150 seconden). Dit kan ongedaan gemaakt worden door in te typen SCREENSAVE OFF of SCREENSAVE 0.

AANVERWANTE COMMANDO'S: SCREENON, SCREENOFF

SET DRIVE *Selecteer een diskdrive.*

SYNTAX: SET DRIVE [drivenaam]

Drivenaam is de gewenste diskdrive: een der letters A,B,C,D,E,F,G of H. Indien disk niet opgegeven wordt, volgt een vraag welke drive geselecteerd moet worden.

WERKING:

Hiermee kunt u vanuit BASIC een bepaalde diskdrive selecteren. Indien geen drive opgegeven wordt, volgt een melding welke drives mogelijk zijn, waarna een keuze gedaan moet worden.
De geselecteerde drive is degene die gebruikt wordt als bij saven en laden niet expliciet een drivenaam toegevoegd wordt.

(FOUT)MELDINGEN:

Illegal function call - de opgegeven drive is niet aangesloten

gebruiksvoorbeeld:

```
SETDRIVE C: DFILES
```

STATUS *Toon een overzicht van de door BASIC gebruikte ruimte.*
LSTATUS *Print een overzicht van de door BASIC gebruikte ruimte.*

SYNTAX: STATUS
LSTATUS

WERKING:

Geeft een overzicht van de gebruikte geheugenruimte, de lengte van het BASIC programma, de vrije geheugenruimte, en het aantal BASIC regels, in de onderstaande vorm:

```
BASIC STATUS
Delta Basic version ..
Copyright 1987 by Filosoft

. Drive           .. Bytes free

PROGRAM    :      .. Bytes
VARS       :      .. Bytes
ARRAYS     :      .. Bytes
STRINGS    :      .. Bytes

BASIC program      .. LINES
from LINE  .. to LINE ..
```

STORE SCREEN *Bewaar beeldscherm in geheugen.*
RESTORE SCREEN *Haal beeldscherm terug uit geheugen.*

SYNTAX: STORE SCREEN
RESTORE SCREEN

WERKING:

Slaat het huidige scherm op in het geheugen, en onthoudt daarbij de screenmode en width. Het scherm kan weer teruggehaald worden d.m.v. het RESTORESCREEN-commando. De complete eerste 16K van de video-RAM worden in het geheugen opgeslagen. Dit is voldoende voor screen 0,1,2 en 3 (inclusief SPRITES) maar niet voor hogere screens (MSX2).
Na een STORESCREEN kan meermalen een RESTORESCREEN gebruikt worden om hetzelfde beeld terug te krijgen.
Het is niet nodig voor een RESTORESCREEN eerst de schermmode in te stellen, omdat deze mee bewaard wordt.

(FOUT)MELDINGEN

MEMDISC initialised - de memorydisk dreigt overschreven te worden; een memorydisk is opgezet en heeft het opgeslagen beeld overschreven

TIPS:

Het geheugendeel waarin het beeldscherm opgeslagen wordt, wordt ook gebruikt voor het commando CHAIN en alle memory-disk commando's.

gebruiksvoorbeeld:

```
10 SCREEN 2:COLOR 15,1,1:KEY OFF
20 '
30 FOR Y=1 TO 5
40 FOR X=15 TO 2 STEP-1
50 CIRCLE ((X-1)*16,Y*32),X,X
60 PAINT ((X-1)*16,Y*32),X
70 NEXT X
80 NEXT Y
90 '
100 STORE SCREEN
110 '
120 FOR N=5 TO 1 STEP-1
130 SCREEN 0
140 PRINT"HET ZOJUIST GEMAAKTE SCHERM IS OPGESLAGEN"
150 PAUSE N
160 RESTORE SCREEN
170 PAUSE N
180 NEXT N
```

TFILES *Toon overzicht bestanden op cassette.*
LTFILES *Print overzicht bestanden op cassette.*

SYNTAX: TFILES [,A]
LTFILES [,A]

Indien de optie ',A' toegevoegd wordt, wordt tevens de inhoud van een bestand op het scherm getoond, indien het een ASCII-bestand betreft.

WERKING:

Geeft informatie over de aanwezige bestanden op een cassette. Alle standaard MSX-bestanden worden herkend.
Het commando blijft in werking totdat CTRL+STOP gedrukt wordt.

Uit de 'headers' van bestanden wordt informatie gegeven over het soort bestand en de naam.
Tevens wordt, afhankelijk van het soort bestand, relevante informatie gegeven over lengte en aantal blokken, dan wel over beginadres, eindadres en opstartadres.

Indien gekozen is voor de optie ',A' wordt van een ASCII-bestand tevens de inhoud weergegeven op het scherm of de printer.

AANVERWANTE COMMANDO'S: FILES, DFILES

TIPS:

Indien eventueel de BIPRINT-mode was ingeschakeld, wordt deze automatisch uitgezet om vertragingen te voorkomen.
Zeker bij gebruik van LTFILES verdient het aanbeveling gebruik te maken van de zogenaamde REMOTE CONTROL mogelijkheid indien de recorder deze kent (zie gebruiksaanwijzing computer).

UNNEW *Haal een (per ongeluk) gewist programma terug.*

SYNTAX: UNNEW

WERKING:

Een programma dat per ongeluk gewist is door een NEW-commando kan met UNNEW worden teruggehaald. De melding "Program recovered" zal op het scherm verschijnen.
Gebruik deze optie alleen na een NEW.
Indien de RESET-toets op de computer gebruikt wordt, zal niet alleen DELTA BASIC in het geheugen blijven, doch zal ook automatisch een UNNEW worden uitgevoerd: een tevoren aanwezig programma blijft aanwezig.

AANVERWANTE COMMANDO'S: NEW

(FOUT)MELDINGEN:

Program recovered - het gewiste programma is teruggehaald
UNNEW without NEW - er was geen programma gewist

UPPER *Zet kleine letters om in hoofdletters.*

SYNTAX: UPPER string

String is een rijvariabele of lijstvariabele.

WERKING:

Dit commando verandert voorkomende kleine letters in een string in hoofdletters.

AANVERWANTE COMMANDO'S: LOWER, CAPSON, CAPSOFF

TIPS:

Upper is een commando, en geen functie. Een expressie als A$=UPPER B$ is dus niet mogelijk!
De omzetting vindt plaats binnen de aangegeven string.

gebruiksvoorbeeld:

```
10 SCREEN 0
20 CAPS OFF
30 INPUT"TYP EEN WOORD EN DRUK <RETURN>";W$
40 PRINT"U TYPTE:";W$
50 UPPER W$
60 PRINT"DE FUNKTIE 'UPPER' MAAKT HIERVAN: ";W$
```

WBOX # *Zet kader rondom een window.*

SYNTAX: WBOX #window

Window is het nummer van een tevoren gedefinieerd window, waarbij moet gelden: 1 <= window <= 9.

WERKING:

Zet een rechthoekig kader rondom het aangegeven window.

(FOUT)MELDINGEN:

```
INVALID definition window w  - window w is verkeerd gedefinieerd
INVALID window               - window nummer niet correct (0<=w<=9)
Illegal function call        - alleen screenmode 0,1,2 toegestaan
MISSING definition window w  - window w is niet gedefinieerd
```

TIPS:

Het omkaderen van een window kan beperkend werken op het gebruik van een WINPUT-commando, aangezien een lijndeel aan de rechterkant altijd beschouwd zal worden als behorend tot de input (niet in SCREEN 2).

gebruiksvoorbeeld:

```
10 SCREEN 0
20 WINDOW #1,3,3,10,10
30 WBOX #1
```

WINDOW # *Definieer een window.*

SYNTAX: WINDOW #w,x,y,x1,y1

W is het windownummer tussen 1 en 9.
X en Y zijn de coördinaten van de windowhoek linksboven.
X1 en Y1 zijn de coördinaten van de windowhoek rechtsonder.
X en X1 zijn minimaal 0 en maximaal (ingestelde) WIDTH.
Y en Y1 zijn minimaal 0 en maximaal 22 (bij KEY ON) of 23 (KEY OFF).
X1 is groter dan of gelijk aan X, en Y1 is groter dan of gelijk aan Y.

WERKING:

Met dit commando kunnen de linkerbovenhoek, en de rechterbenedenhoek van een bepaald window worden gedefinieerd. De toegestane waarden zijn afhankelijk van screenmode, width en KEY ON/OFF, zowel bij de definitie van een window, als bij uitvoering van elke functie die op dat window betrekking heeft.
In SCREEN 2 worden de X- en Y-waarden geteld in blokjes van 8 pixels; hier is dus de maximale X-waarde 32 en de maximale Y-waarde 23.
Bij verandering van screenmode blijven de window-definities gehandhaafd. Het is van belang hierbij de toelaatbare waarden in het oog te houden.

Window #0 is standaard gedefinieerd als zijnde het gehele beeldscherm. Dit window hoeft en kan niet apart gedefinieerd worden. Wel zijn alle windowfuncties hierbinnen mogelijk (bijvoorbeeld ROLL en FILL).

Alle window-commando's werken uitsluitend in de screenmodes 0, 1 en 2.

AANVERWANTE COMMANDO'S: CLS #, FILL #, LOCATE #, WBOX #
WPRINT #, WINPUT #, ROLL #, WRAP #,
(L)LIST WINDOW

(FOUT)MELDINGEN:

INVALID definition window w	- window w is verkeerd gedefinieerd
INVALID window	- window nummer niet correct (0<=w<=9)
Illegal function call	- alleen screenmode 0,1,2 toegestaan!

TIPS:

Zorg ervoor dat het betreffende window zorgvuldig gedefinieerd wordt. Veranderingen van screenmode of width, en zelfs KEYON of KEYOFF kunnen een window definitie ongeldig maken.

gebruiksvoorbeeld:

```
10 SCREEN 2
20 WINDOW #1,1,1,31,4
30 WINDOW #2,1,7,15,13
40 WINDOW #9,1,22,31,22
```

WINPUT # *Input binnen een window.*

SYNTAX: WINPUT #window;["tekst"];var

Window is het nummer van een gedefinieerd window (tussen 1 en 9).
Tekst wordt op de plaats waar de invoer verwacht wordt, op het scherm gezet, voorafgaand aan een vraagteken; de tekst moet tussen aanhalingstekens in het programma staan.
Var is een variabele (alle soorten mogelijk).

WERKING:

Een WINPUT-commando is vergelijkbaar met het normale INPUT-commando, met dit verschil dat het geheel binnen de grenzen van het aangegeven window blijft.
Bovendien dient bij de uitvoering van het programma de gevraagde input op een regel te blijven; ook mag er geen kader of tekst van een ander window rechts van de input voorkomen.

AANVERWANTE COMMANDO'S: WINDOW #, LOCATE #, WPRINT #, INPUT

(FOUT)MELDINGEN:

```
INVALID definition window w - window w is verkeerd gedefinieerd
INVALID window              - window nummer niet correct (1<=w<=9)
Illegal function call       - alleen screenmode 0,1,2 toegestaan!
MISSING definition window w - window w is niet gedefinieerd
```

TIPS:

Zorg ervoor dat het betreffende window juist gedefinieerd is. Veranderingen van screenmode of width, en zelfs KEYON of KEYOFF kunnen een window definitie ongeldig maken.
Net als de andere window-commando's werkt het WINPUT-commando ook in SCREEN 2.
Indien gebruik is gemaakt van een WBOX-commando, of een gevuld window bestaat rechts van het window waarin de WINPUT plaats vindt, wordt daarvan een en ander meegenomen in de INPUT. Het daarna gebruiken van de functie LEFT kan hierbij een oplossing bieden.

gebruiksvoorbeeld

```
10 SCREEN 0:KEY OFF:WIDTH 40
20 WINDOW #2,3,4,35,15
30 FOR N=1 TO 30
40 WPRINT #2;"DIT IS WINDOW 2";
50 NEXT N
60 WINPUT #2;"TYP EEN WOORD EN <RETURN>";W$
70 WPRINT #2:WPRINT #2;"HET WOORD WAS:";W$
```

WPRINT # *Geef weer binnen een window.*

SYNTAX: WPRINT #w;uitdrukking[;uitdrukking;.....]

Uitdrukking is een letterlijke tekst tussen aanhalingstekens, een getal of een variabele.
Meerdere uitdrukkingen kunnen achter één WPRINT-commando volgen, mits onderling gescheiden door een puntkomma.

WERKING:

Hiermee kan op normale wijze op het scherm geprint worden, waarbij de tekst binnen het opgegeven window zal blijven. Wanneer het window "vol" is zal het naar boven rollen om ruimte te maken voor nieuwe tekst.

AANVERWANTE COMMANDO'S: WINDOW #, LOCATE #, WINPUT #, PRINT

(FOUT)MELDINGEN:

```
INVALID definition window w - window w is verkeerd gedefinieerd
INVALID window              - window nummer niet correct (0<=w<=9)
Illegal function call       - alleen screenmode 0,1,2 toegestaan!
MISSING definition window w - window w is niet gedefinieerd
```

TIPS:

Zorg ervoor dat het betreffende window juist gedefinieerd is. Veranderingen van screenmode of width, en zelfs KEYON of KEYOFF kunnen een window definitie ongeldig maken.

gebruiksvoorbeeld:

```
10 SCREEN 2
20 WINDOW #9,24,0,31,0
30 ON INTERVAL=50 GOSUB 9000
40 T=0:INTERVAL ON
.
.
9000 T=T+1:WPRINT #9;T:RETURN
```

WRAPD # *Rol inhoud window omlaag met omslag.*
WRAPL # *Rol inhoud window naar links met omslag.*
WRAPR # *Rol inhoud window naar rechts met omslag.*
WRAPU # *Rol inhoud window omhoog met omslag.*

SYNTAX: WRAPD #window[,aantal]
WRAPL #window[,aantal]
WRAPR #window[,aantal]
WRAPU #window[,aantal]

Window is het nummer tussen 0 en 9 van een tevoren gedefinieerd window. Windownummer 0 (nul) laat het gehele scherm rollen.
Aantal is het aantal keren dat direct achter elkaar gerold dient te worden; aantal is een geheel getal tussen 1 en 255. Aantal is een optie en mag ook weggelaten worden (hetzelfde als aantal=1).

WERKING:

Rolt de inhoud van een window (tekst of grafiek) het aantal opgegeven posities in de gewenste richting. De regel of kolom die in de opgegeven richting buiten het window verdwijnt, wordt aan de tegenovergestelde kant weer toegevoegd.
Een positie rollen betekent in SCREEN 2 dat 8 pixels tegelijk geschoven wordt.

AANVERWANTE COMMANDO'S :ROLLD #, ROLLL #, ROLLR #, ROLLU #, WINDOW #

(FOUT)MELDINGEN :

INVALID definition window w - window w is verkeerd gedefinieerd
INVALID window - windownummer niet correct (0<=W<=9)
Illegal function call - alleen screenmode 0,1,2 toegestaan!
MISSING window definition w - window w is niet gedefinieerd

TIPS :

Zorg ervoor dat het betreffende window juist gedefinieerd is. Veranderingen van screenmode of width, en zelfs KEYON of KEYOFF kunnen een windowdefinitie ongeldig maken.

gebruiksvoorbeeld:

```
10 SCREEN 0:KEY OFF:WIDTH 40
20 WINDOW #1,1,1,38,10
30 WINDOW #2,1,13,18,20
40 WINDOW #3,23,13,38,20
50 FOR W=1 TO 3:WBOX #W:FOR N=1 TO 30
60 WPRINT #W;"DIT IS WINDOW";W;:NEXT N:NEXT W
70 WRAPU #1,10
80 WRAPL #2,10
90 WRAPR #3,10
100 WRAPD #0,24
```

BIJLAGE A - BIT IMAGE PRINTEN

Delta Basic biedt speciale mogelijkheden voor het zogenaamde bit-image printen. Wellicht ten overvloede: u kunt hiervan alleen gebruik maken als u beschikt over een dot-matrix printer; alle letterwiel-printers en de meeste plotters kennen deze mogelijkheid niet!

De waarden die achter een BISET-commando moeten worden ingegeven, kunnen getypt worden als decimale en hexadecimale getallen, als variabelen of als uitkomsten van functies.
Bijvoorbeeld: A=&H1B: BISET(A,ASC("*"),0,480 MOD 256,480/256) ; dit is hetzelfde als: BISET(27,42,0,224,1).

Hieronder staat een tabel met codes in decimale waarden, die voor veelgebruikte printers gelden. Helaas kunnen we niet uitputtend zijn; ook zijn voor genoemde printers soms andere codereeksen bruikbaar. Met name de codes voor de regelafstand die het beste resultaat geeft bij een screendump, kan wel eens afwijken. Experimenteer om het beste resultaat te krijgen.

ALGEMEEN ADVIES: BESTUDEER UW PRINTER-HANDLEIDING!

printersoort	bit-image printen	korte regelafst.	norm.afst.	upside
MSX-printers	27,83,48,52,56,48	27,66	27,65	U
EPSON	27,42,0,224,1	27,49	27,50	-
BROTHER	27,75,224,1	27,49	27,50	-
STAR	27,75,224,1	27,65,8,27,50	27,50	-
NEC/C-ITOH	27,83,48,52,56,48	27,84,49,53	27,65	U

De codes voor korte regelafstand zijn die, welke voor een screendump een verticaal aansluitend beeld geven; onder 'norm.afst.' worden de codes gegeven die de regelafstand weer normaal zet.
Onder upside wordt aangegeven of het nodig is bij het commando BIPRINT de optie ',U' te gebruiken.

BIJLAGE B - (FOUT)MELDINGEN DELTA BASIC

200 - ENDPROC without PROC
Er wordt een ENDPROC gevonden zonder dat er een PROC-opdrach is geweest; nesten van procedures is niet toegestaan.

201 - DEFPROC not found
De procedure met de opgegeven naam kan niet worden gevonden, of de procedurenaam is niet door aanhalingstekens afgesloten.

202 - ENDPROC not found
De procedure wordt niet correct afgesloten.

203 - MODE error
De opdracht mag niet in een programma voorkomen, alleen als direkt commando bruikbaar; of: alleen in programma te gebruiken

204 - UNNEW without NEW
Er is geen programma gewist; er is geen programma geweest.

205 - Program recovered
Het gewiste programma is weer teruggehaald.

206 - TOO MUCH varspace
Er is geen ruimte voor alle met COMMON aangewezen variabelen.

207 - NO ROOM for vars
Na het inladen van het nieuwe programma is er niet genoeg ruimte om alle variabelen terug te halen.

208 - INVALID definition window
Window onjuist gedefinieerd; bekijk grenzen opnieuw (LISTWINDOW)

209 - INVALID window
Alleen window-nummers tussen 0 en 9 toegestaan.

210 - MISSING definition window
Het bedoelde window is nog niet gedefinieerd.

211 - LOCATE out of window
Een LOCATE#-opdracht valt buiten de grenzen van het window.

212 - MEMDISC initialised
De memory-disk dreigt overschreven te worden.

213 - LINE number too big
Een regelnummer groter dan 65529 dreigt te ontstaan.

214 - MOVLIN error
Een der opgegeven waarden is onjuist.

215 - COPLIN error
Een der opgegeven waarden is onjuist.

216 - STATE size
Bij het installeren van de memory-disk moet verplicht de omvang worden opgegeven (maximaal 16000 bytes)

DELTA BASIC

Hoezo slechts 24 K beschikbaar voor Basic? Met DELTA BASIC maakt u programma's van 100 tot 10.000 K in Basic!

Waarom nog verdwalen in een bos van GOSUB-routines? DELTA BASIC geeft uw MSX-computer de mogelijkheid van PROCEDURES, waarvan u zelf een in principe oneindige bibliotheek kunt aanleggen!

Is uw beeldscherm simpel? Met DELTA BASIC beschikt u over tien verschillende WINDOWS!

DELTA BASIC verlegt de grenzen van MSX-BASIC. DELTA BASIC kent het gebruik van WINDOWS voor een overzichtelijker en efficiënter beeldschermgebruik, de CHAIN-mogelijkheid om vanuit een programma nieuwe programma's in te laden met behoud van bestaande variabelen, het gebruik van PROCEDURES voor beter en inzichtelijker programmeren, en een schat aan extra utilities waarover elke programmeur reeds lang wilde beschikken.

WINDOWS

- **WINDOW** (definieert window; max. 10 tegelijk mogelijk)
- **LOCATE** (zet cursorpositie binnen window; 10 onzichtbare cursors!)
- **WPRINT** (geeft weer binnen window)
- **WINPUT** (vraagt input binnen window)
- **WBOX** (zet kader om window)
- **CLS** (veegt window schoon)
- **FILL** (vult window met lettertekens)
- **ROLL** (rolt inhoud window naar boven/onder/links/rechts)
- **WRAP** (idem; wat aan ene kant verdwijnt, komt aan andere kant terug)

CHAIN

- **COMMON** (legt te bewaren variabelen vast)
- **CHAIN** (laadt en runt ander programma, en haalt variabelen terug)

PROCEDURES

- **DEFPROC** (geeft begin procedure aan)
- **ENDPROC** (geeft eind procedure aan)
- **PROC** (roept procedure aan)
- **SAVEPROC** (bewaart procedure op disk/cass.)
- **MERGEPROC** (haalt procedure van disk/cass. en koppelt 'm aan programma)
- **DELETEPROC** (wist procedure)

PLUS...

- **TFILES/LTFILES** (overzicht bestanden op cassette)
- **AVERIFY** (verifieert BASIC-ASCII bestand op cassette)
- **BVERIFY** (verifieert byte-bestand op cassette)
- **BAUD** (selecteert BAUD-rate voor cassette-saven tussen 900 en 3000)
- **DFILES/LDFILES** (geeft inhoud disk inclusief lengtes en vrije ruimte)
- **SETDRIVE** (selecteert default drive)
- **STORESCREEN/RESTORESCREEN** (bewaart/haalt compleet scherm terug)
- **SCREENSAVE** (maakt scherm na bepaalde tijd donker tegen inbranden)
- **SCREEN/OFF** (zet scherm aan/uit)
- **CLEARSPRITES** (wist alle sprites)
- **INIPSG** (initieert sound-generator, stopt geluid direct)
- **INIFNK** (zet functietoetsen terug op originele waarden)
- **KILLBUF** (wist toetsenbord-buffer)
- **GET** (pakt eerstvolgende toetsdruk)
- **PAUSE** (wacht bepaalde tijd of tot toetsindruk)
- **CAPSON/OFF** (zet hoofdletter-mode aan/uit)
- **POL** (kijkt of printer on-line staat)
- **BISET/BIPRINT** (selecteert bit-image mode op printer)
- **SCREENDUMP** (stuurt beeldscherm naar printer)
- **DPEEK** (geeft de inhoud van 2 geheugenadressen samen; 0-65536)
- **DPOKE** (zet getal 0-65536 in twee opeenvolgende geheugenadressen)
- **LOWER** (verzekert kleine letters in string)
- **UPPER** (verzekert hoofdletters in string)
- **STATUS/LSTATUS** (geeft overzicht geheugengebruik)
- **UNNEW** (haalt programma terug na NEW)
- **FIND** (zoekt/vervangt tekst in BASIC-programma)
- **MOVLIN** (verplaatst BASIC-regels)
- **COPLIN** (kopieert BASIC-regels)
- **CODETODATA** (zet geheugeninhoud om in DATA-regels)
- **LISTGO/LLISTGO** (overzicht regelnrs. waar GOSUB's/GOTO's heenspringen)
- **LISTLIN/LLIST** (overzicht regelnrs. waarin GOSUB's/GOTO's voorkomen)
- **LISTPROC/LLISTPROC** (overzicht procedures)
- **LISTVAR/LLISTVAR** (overzicht variabelen)
- **LISTTYPE/LLISTTYPE** (overzicht variabelen-typeringen)
- **LISTDATA/LLISTDATA** (overzicht regelnrs. met DATA-statements)
- **LISTUSR/LLISTUSR** (overzicht adressen door DEFUSR's aangegeven)
- **LISTPSG/LLISTPSG** (overzicht instellingen geluidsgenerator)
- **LISTSOUND/LLISTSOUND** (overzicht actuele sound-waarden)
- **LISTSCREEN/LLISTSCREEN** (overzicht SCREEN-waarden)
- **ETC., ETC.**

DELTA BASIC werkt op alle MSX-computers met tenminste 64K geheugen. Het wordt van tevoren in het geheugen geladen, en neemt slechts enkele bytes van de beschikbare BASIC-ruimte af. Alle beeldscherm-opdrachten werken in SCREEN 0, 1 en 2. DELTA BASIC blijft in het geheugen ook na een RESET. Het programma heeft een uitgebreide Nederlandstalige handleiding, met natuurlijk een syntax-overzicht van alle nieuwe commando's en functies, en tal van voorbeeldprogramma's.